

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

**GRADO EN INGENIERÍA EN ELECTRÓNICA Y
AUTOMÁTICA INDUSTRIAL**

Trabajo Fin de Grado

**DESARROLLO COMUNICACIÓN ALFA ARDUINO
MEDIANTE UN SCADA**

Autor: Jorge Trallero Calvo

Director: Ignacio Bravo Muñoz

TRIBUNAL:

Presidente:

Vocal 1º:

Vocal 2º:

CALIFICACIÓN:

FECHA:

A mi familia

“Un hombre que no pasa tiempo con su familia nunca puede ser un hombre de verdad”

The Godfather

Agradecimientos

Agradezco principalmente y quiero dedicar este proyecto a mi familia. Ellos han sido los encargados de proporcionar todos y cada uno de los medios necesarios para realizar mis estudios, animándome en los momentos difíciles consiguiendo que supere cada uno de los obstáculos que me he encontrado en mi vida, sabiendo comprender y aguantar todos y cada uno de mis problemas, solo puedo decir GRACIAS por todos los momentos que he disfrutado con todos vosotros.

Agradecer a Antonio todo el apoyo recibido, porque “hoy puede ser un gran día”, frase que siempre me ha acompañado día a día.

También quiero agradecer a todos y cada uno de los amigos que han estado fuera y dentro de la Universidad, por todos los buenos y malos momentos que hemos vivido juntos esperando poder disfrutar más momentos con todos vosotros y aprender de cada uno de vosotros.

Por último agradecer a Ignacio, tutor de mi proyecto, la ayuda proporcionada y su tiempo para poder terminar este proyecto de manera satisfactoria.

A todos y cada uno de vosotros por vivir esta etapa de mi vida conmigo, que con este proyecto finalizo esperando poder seguir teniendo a todos en la próxima, muchas gracias por todo lo recibido.

Contenido

1	Resumen.....	15
2	Summary.....	17
3	Palabras Clave.....	19
4	Resumen extendido	21
5	Introducción.....	25
5.1	Objetivo principal.....	25
5.2	Objetivos secundarios	26
5.3	Objetivos personales.....	26
6	Estado del Arte.....	27
6.1	Sofia2.....	27
6.2	eDiana Acciona	28
6.3	Libelium certified cloud partner	29
7	IoT (Internet of Things).....	31
7.1	Historia	31
7.2	FI-WARE	32
8	Arduino.....	33
8.1	Historia	33
8.2	¿Qué es y cómo funciona?	34
8.3	Puntos a favor de Arduino:.....	34
8.4	Puntos en contra de Arduino:.....	34
9	Herramientas utilizadas.....	37
9.1	Hardware.....	37
9.1.1	Arduino MEGA 2560.....	37
9.1.2	Entradas/Salidas (I/O)	38
9.1.3	Arduino Ethernet Shield:.....	39

9.1.4	Sensor Waterproof DS18B20 Digital Temperature	40
9.1.5	Buses de datos	42
9.2	SOFTWARE	43
9.2.1	C++/C#	43
9.2.2	Arduino 1.0.5-r2	44
9.2.3	LabView 2012	44
10	Diseño realizado	47
10.1	Base teórica	48
10.1.1	Telemedida	48
10.1.2	Comunicación TCP/IP:	48
10.1.3	Adquisición de datos	52
10.1.4	SCADA LabView	53
10.2	Base práctica	55
10.2.1	Pasos previos	56
10.3	Desglose TOP-DOWN	65
10.3.1	Arduino cliente	65
10.3.2	Arduino maestro	66
10.3.3	SCADA LabView	68
10.4	Arquitectura completa	71
10.5	Problemas y soluciones	76
11	Conclusiones y trabajos futuros	80
11.1	Suprimir LabView	81
11.2	Implementar actuadores	81
11.3	Base de datos MongoDB y Hadoop	81
12	Diagramas	83
12.1	Main_maestro	83

12.1.1	Lectura():.....	83
12.1.2	Newclient():	84
12.1.3	LabView():	84
12.2	Main_cliente.....	84
12.2.1	Temperatura():	84
12.2.2	Envio_datos():	85
12.3	SCADATEMPERATURA.vi.....	85
13	Pliego de condiciones.....	87
13.1	Objetivo de este documento	87
13.2	Condiciones generales	87
13.2.1	Documentos que rigen la ejecución de obra	87
13.2.2	Condiciones generales de pago y entrega del trabajo	87
13.3	Condiciones de materiales y equipos	88
13.3.1	Equipos de conexión	88
13.3.2	Equipos de programación.....	89
13.4	Condiciones de ejecución	89
14	Presupuesto	91
14.1	Software.....	91
14.1.1	Listado de software necesario:.....	91
14.2	Hardware	91
14.2.1	Listado de hardware necesario:	91
14.3	Recursos.....	92
14.4	Mano de obra.....	92
14.5	Presupuesto de ejecución material	93
14.6	Presupuesto de ejecución por contrata.....	93
14.7	Presupuesto total.	94

15	Montaje y manual de usuario.....	95
15.1	Introducción	95
15.2	Recursos asociados a este documento	95
15.3	Configuración de inicio.....	96
15.3.1	Configuración maestro	98
15.3.2	Configuración cliente	100
15.3.3	Conexión sensor de temperatura	101
15.4	Arranque del sistema	102
16	Bibliografía.....	105

Índice de Figuras

Fig. I Esquema nivel básico funcionamiento de Sofia2	28
Fig. II Diagrama de funcionamiento de libelium certified cloud partner	29
Fig. III Arduino Mega 2560	37
Fig. IV Arduino Ethernet Shield	39
Fig. V Ethernet module.....	40
Fig. VI Sensor DS18B20	41
Fig. VII Cable Ethernet con conexión RJ45.....	42
Fig. VIII USB 2.0 tipo A-B	42
Fig. IX Símbolo USB	43
Fig. X Logotipo LabView	45
Fig. XI Esquema diseño realizado.....	47
Fig. XII Capas TCP/IP	49
Fig. XIII Relación OSI-TCP/IP	50
Fig. XIV Inicialización Servidor-Cliente.....	51
Fig. XV Comunicación servidor-cliente.....	52
Fig. XVI Diagrama de flujo del establecimiento de conexión en el sistema activo.	54
Fig. XVII Diagrama de flujo de una operación de lectura o recepción de datos TCP/IP.....	55
Fig. XVIII diseño inicial	56
Fig. XIX Resultado diseño inicial	56
Fig. XX estructura while	57
Fig. XXI Conexión tres clientes	59
Fig. XXII Comunicación entre Arduinos.....	59
Fig. XXIII Diagrama de flujo nuevo cliente	61

Fig. XXIV Respuesta Arduino Maestro-nuevo cliente.....	62
Fig. XXV Respuesta Arduino cliente-nuevo cliente	62
Fig. XXVI LabView Lectura Ethernet nº clientes.....	63
Fig. XXVII Panel frontal	64
Fig. XXVIII Arduino cliente.....	65
Fig. XXIX Diagrama Arduino cliente	66
Fig. XXX Arduino maestro	67
Fig. XXXI Diagrama Arduino Maestro	67
Fig. XXXII SCADA LabView	68
Fig. XXXIII Diagrama SCADA LabView.....	69
Fig. XXXIV TCP Open Connection	69
Fig. XXXV TCP Write	70
Fig. XXXVI TCP Read	70
Fig. XXXVII TCP Close	71
Fig. XXXVIII SCADA LabView.....	71
Fig. XXXIX Puerto serie Arduino. 0 clientes	72
Fig. XL Comunicación fallida.....	72
Fig. XLI Reacción Arduino cliente	73
Fig. XLII Reacción Arduino maestro.....	73
Fig. XLIII frontal LabView	74
Fig. XLIV Frontal LabView 4 sensores activos	74
Fig. XLV Comunicación cliente a cliente	75
Fig. XLVI Comunicación LabView	75
Fig. XLVII Resultado SCADA LabView	76
Fig. XLVIII Separación parte decimal-parte entera.....	78
Fig. XLIX Flat Sequence Lectura.....	79

Fig. L void loop Maestro	83
Fig. LI void loop Cliente	84
Fig. LII Diagram & Front LabView	85
Fig. LIII Entorno de desarrollo Arduino	96
Fig. LIV Elección Tarjeta Arduino	97
Fig. LV Elección Puerto de Conexión	97
Fig. LVI Carga código Arduino	98
Fig. LVII Configuración Maestro, paso 1	98
Fig. LVIII Configuración Maestro, paso 2	99
Fig. LIX Configuración Maestro, paso 3	99
Fig. LX Configuración Cliente, paso 1	100
Fig. LXI Configuración Cliente, paso 2	100
Fig. LXII Configuración Cliente, paso 3	101
Fig. LXIII Conexión Sensor-Arduino	101
Fig. LXIV Conexión Arduino Mega	102
Fig. LXV Programa de visualización	103
Fig. LXVI Programa de visualización, 1 cliente	103

1 Resumen

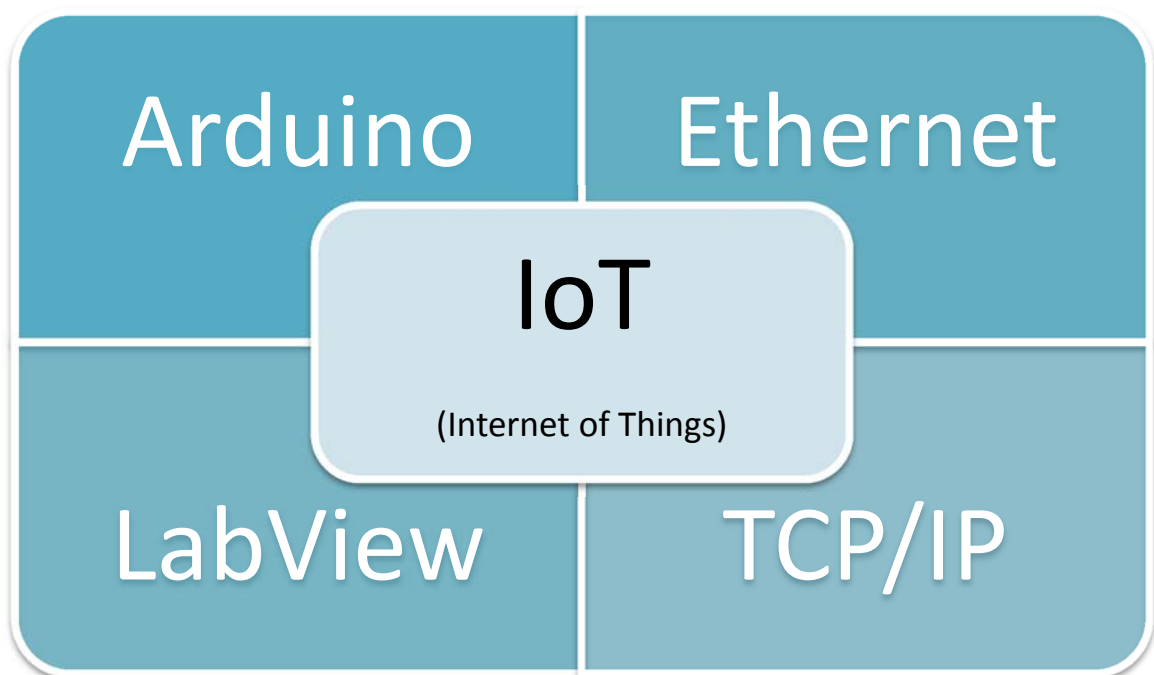
El siguiente proyecto pretende crear un sistema IoT (Internet of Things) girando en torno a Arduino, demostrando de este modo la facilidad de creación de proyectos y fiabilidad que nos aporta Arduino. Mediante una comunicación TCP/IP se procede a crear un sistema el cual es capaz de registrar valores a una cierta distancia mediante protocolos de procesos y comunicación (Telemedida) y de este modo monitorizar la temperatura en un punto determinado, sin estar limitado por la teniendo el dispositivo conectado a Internet y controlarlo a partir de un dispositivo como ordenadores, teléfonos móviles, tablets... mediante una aplicación en LabView.

2 Summary

This new project aims to create a new IoT system in Arduino, pretending to prove the nearly limitless array of innovative applications and reliability of Arduino based projects.

By using TCP/IP networking protocols, a new system capable of registering remote input values through communication protocols (telemetry) and by doing so, remotely measure the temperature of a motorized point. Through Internet connection we can control it from our computer, smartphone, tablet... with a LabView application.

3 Palabras Clave



4 Resumen extendido

El Internet de las Cosas es un concepto difícil de definir con precisión. De hecho, hay muchos grupos diferentes que han definido el término, aunque su uso inicial se ha atribuido a Kevin Ashton, experto en innovación digital. Cada definición comparte la idea de que la primera versión de Internet se definía para los datos producidos por las personas, mientras que la próxima versión trata sobre los datos creados por las cosas. En 1999, Ashton cita en un artículo del Diario de RFID: "Si tuviéramos equipos que supiesen todo lo que habría que saber sobre las cosas (a partir de datos que se reúnen sin la ayuda de nosotros) que fuese capaz de hacer un seguimiento y controlarlo todo, y reducir considerablemente los residuos, las pérdidas y el costo. Nos gustaría saber cuándo las cosas necesitan ser reemplazadas, reparadas o retiradas del mercado". La mayoría de nosotros piensa en estar conectados en términos de ordenadores, tablets y Smartphone. IoT describe un mundo en el que casi cualquier cosa puede conectarse y comunicarse de una manera inteligente. En otras palabras, con "el Internet de las cosas", el mundo físico se está convirtiendo en un sistema de información grande. Referenciando a los trabajos previos ya existentes como Sofia2, eDiana... enfoco de manera primaria estas ideas, llegando a un nivel de utilización básica para el cliente, sin necesidad de saber crear una aplicación para poder llegar a obtener una funcionalidad de un sistema implementado, como ocurre con Sofia2, donde un usuario puede proceder a la creación de un programa con el cual, dependiendo de su enfoque, le proporcionara una información determinada.

Con este proyecto pretendo romper las barreras entre los sistemas y el usuario, procediendo a crear un sistema simple y capaz de ser controlado y entendido a un nivel de informática mínimo, sin necesidad de ser un programador, o entender todos los sistemas y parámetros de su entorno.

La finalidad consiste en la monitorización de un edificio, sala o entorno, mediante la utilización de sistemas empotrados con un coste reducido y fáciles de sustituir si llega a ser necesario, y mediante un interfaz creado para la visualización de las medidas recogidas de una manera sencilla y práctica.

El proyecto propuesto “Desarrollo comunicación Alfa Arduino mediante un SCADA” encontramos un objetivo principal, que es ofrecer al usuario medio la posibilidad de poder manejar dispositivos con un fin comercial, empresarial o privado, la monitorización y control de un entorno específico.

Cuando hablamos de IoT, tenemos que tener en cuenta al alto crecimiento que ocurre año tras año sobre el número de dispositivos conectados a Internet, superando con creces a los usuarios conectados a la red, de este modo podemos hablar que en un futuro cercano cualquier cosa estará conectada a la nube, registrando datos, los cuales estarán a disposición del usuario. A día de hoy encontramos grandes proyectos sobre Smart Cities, IoT ... encargados de estos proyectos grandes empresas, las cuales sus potenciales clientes son ayuntamientos, otras empresas, grandes empresarios... dejando un mercado sin explotar, que es donde quiero llegar con este proyecto.

Como la finalidad del proyecto es recoger datos provenientes de sensores remotos mediante una comunicación Ethernet entre un ordenador (utilizando un SCADA) y una placa Arduino MEGA, siendo un microcontrolador Atmega1280 (Atmel megaAVR) con puertos de entrada/salida, salidas digitales y entradas digitales/analógicas.

Principalmente con el ordenador tendremos acceso al Arduino MEGA, el cual lo utilizaremos como un GATEWAY (puerta de enlace) existiendo una comunicación con indefinidos nodos, los que se encontraran a una distancia ilimitada de la puerta de enlace, pudiendo de este modo monitorizar indefinidos sistemas y poder tener acceso a la información recogida con una única comunicación con un microprocesador (ATmega1280).

El primer paso en este proyecto es la creación del sistema comentado anteriormente utilizando la plataforma libre de Arduino con la que podremos generar un sistema de manera económica al monitorizar varias naves de una farmacéutica donde se guardan medicamentos con la obligación de mantener una temperatura determinada y mantener de manera satisfactoria los medicamentos.

La explicación de la posibilidad de poder crear un sistema económico es la utilización de una plataforma libre, basada en una placa con un microcontrolador y un entorno de desarrollo, con el que tendremos una facilidad del uso de la electrónica en proyectos multidisciplinarios.

El primer punto será la creación de un script mediante el lenguaje propio de Arduino basado en el lenguaje de alto nivel Processing (código abierto basado en Java) donde implementamos sobre nuestra placa principal de Arduino Mega una comunicación Ethernet entre dos dispositivos, un SCADA creado mediante la utilización de LabView y la comunicación entre la placa Arduino y los nodos que utilizaremos para recopilar los datos medidos de temperatura. Tras realizar este punto primordial se configura los nodos, creando la comunicación mediante Ethernet realizando un intercambio de información con la puerta de enlace, controlando cada nodo mediante la dirección MAC o dirección física, (siendo un identificador de 48 bits correspondiendo de forma única para cada dispositivo) y poder en este caso identificar los datos de cada nodo.

Al utilizar una placa Arduino como puerta de enlace, siendo utilizada os al mismo tiempo como otro sistema independiente, con el cual podremos realizar diversas tareas de lectura o escritura, controlando otra tarea independientemente de los nodos.

Respecto a la representación de todos los datos obtenidos, lo realizaremos como ya hemos comentado, con un SCADA, que será realizado mediante el programa LabView 2012, con el cual estableceremos una comunicación de red utilizando protocolos TCP/IP y mostraremos los datos recopilados mediante los sensores de temperatura utilizados para la finalidad del proyecto. Este programa tendrá que ser capaz de comunicarse únicamente con la placa utilizada como GATEWAY (Arduino MEGA) y representar la información obtenida.

Con la descripción general del proyecto, lo que queremos conseguir es abaratar y minimizar la complejidad de un sistema de control de temperatura, teniendo un gran abanico de posibilidades en el registro de datos y un estudio de los mismos respecto al tiempo, demostrando la fiabilidad del sistema Arduino y la facilidad que nos aporta para realizar proyectos multidisciplinarios.

5 Introducción

El mundo avanza a pasos agigantados en el entorno de la tecnología, hasta tal punto que la máquina puede llegar a controlar otras máquinas y saltarse la intermediación del ser humano, la inteligencia artificial está a la orden del día y adquisición de datos automáticos aumentan día a día. En relación a todo este campo se definió el término IoT “Internet de las cosas”, fundamentado en la conexión a Internet de los dispositivos manteniendo una comunicación autónoma con otros dispositivos sin necesidad del control humano, a partir de esta idea y observando el ámbito empresarial que la rodea he creado un sistema asequible a cualquier bolsillo y un amplio nivel de conocimientos sobre la informática, llegando a un nivel básico del control de la misma, ya que la mayoría de los sistemas creados a día de hoy para la adquisición de datos, monitorización y registro de los mismos están destinados a grandes organizaciones, sin posibilitar la adquisición de tal sistema a pequeñas empresas o a autónomos ya sea por su alto coste, su gran variedad de funcionalidades innecesarias para el objetivo de la pequeña empresa o el autónomo o su complejidad de uso.

En la presente memoria del proyecto, la cual se encuentra estructurada en varios apartados, comenzando con una introducción al entorno que afecta al proyecto, mostrando todos los dispositivos utilizados para tal fin y su utilización, definiendo a continuación la parte teórica de nuestro proyecto y los pasos prácticos seguidos para finalizar satisfactoriamente todo el entorno a desarrollar. Finalizamos la memoria con una guía de instalación para el usuario, un pliego de condiciones y un presupuesto base del proyecto.

Respecto a los objetivos a conseguir en este proyecto, hemos marcados tres apartados diferenciados, como objetivo principal, objetivos secundarios y personales.

5.1 Objetivo principal

El objetivo principal de este proyecto es la creación de un sistema de adquisición de datos, capaz de registrar temperaturas del entorno y poder visualizarlas en un programa generado para ordenador, permitiendo poder realizar un control de varias estaciones de trabajo. En nuestro proyecto,

determinamos el control de almacenamiento de medicamentos para su correcta conservación y mantenimiento, asegurando una adquisición de datos segura y fiable.

5.2 Objetivos secundarios

1-. El estudio de la definición de IoT y su gran importancia a día de hoy, siendo una visión de conectividad que se encuentra en crecimiento y en constante mejora.

2-. El entendimiento del mundo que rodea al término IoT y su influencia en nuestro mundo virtual y la conectividad que conseguimos en el día a día.

3-. La explotación del negocio que rodea al término IoT pudiendo obtener unos grandes beneficios a la hora de enfocar el negocio a medianas y pequeñas empresas, y de forma secundaria a nivel de usuario para cualquier fin que consista en la adquisición de datos de manera automática a partir de un piloto, en nuestro caso un sistema Arduino más un sensor de temperatura con la posibilidad de implementar otro tipo de sensores o actuadores capaces de modificar el ambiente de un entorno.

4-. Abaratar costos utilizando sistemas “Open Source” (código abierto) como la plataforma Arduino, consiguiendo de este modo, la creación de un sistema asequible económicamente y al alcance de más usuarios.

5.3 Objetivos personales

1-. El aprendizaje de la programación en C++, y en LabView, mejorando mi entendimiento en el mundo del desarrollador.

2-. El entendimiento y asentamiento de los conocimientos relacionados con la comunicación TCP/IP entre dispositivos.

6 Estado del Arte

A día de hoy encontramos varias herramientas disponibles a estudiar, en relación con el tema a tratar en este proyecto, donde a partir de trabajos previos existentes nos basamos en gran parte para nuestro fin. Encontramos grandes proyectos sobre Smart Cities, IoT... encargados de estos proyectos grandes empresas, como Indra, Acciona, las cuales sus potenciales clientes son ayuntamientos, otras empresas o grandes empresarios, con proyectos como los siguientes:

6.1 Sofia2



Sofia2 es una Plataforma IoT que actúa como middleware + repositorio, es capaz de procesar miles de eventos por segundo, con capacidades almacenamiento y analítica Big Data, con reglas integradas, interfaces multiprotocolo y multilenguaje y todo operable desde una consola Web.

Es una solución de software integral que transforma productos de uso diario en productos interactivos inteligentes de manera rápida, fácil y económica. La Plataforma Sofia2 combina tecnologías de software y redes innovadoras con las plataformas basadas en la nube y servicios de información digital, y soporta el hardware de los principales proveedores de componentes. Nuestro enfoque permite a los fabricantes integrar sus productos sin modificaciones sustanciales de diseño o cambios en los modelos de negocio existentes.

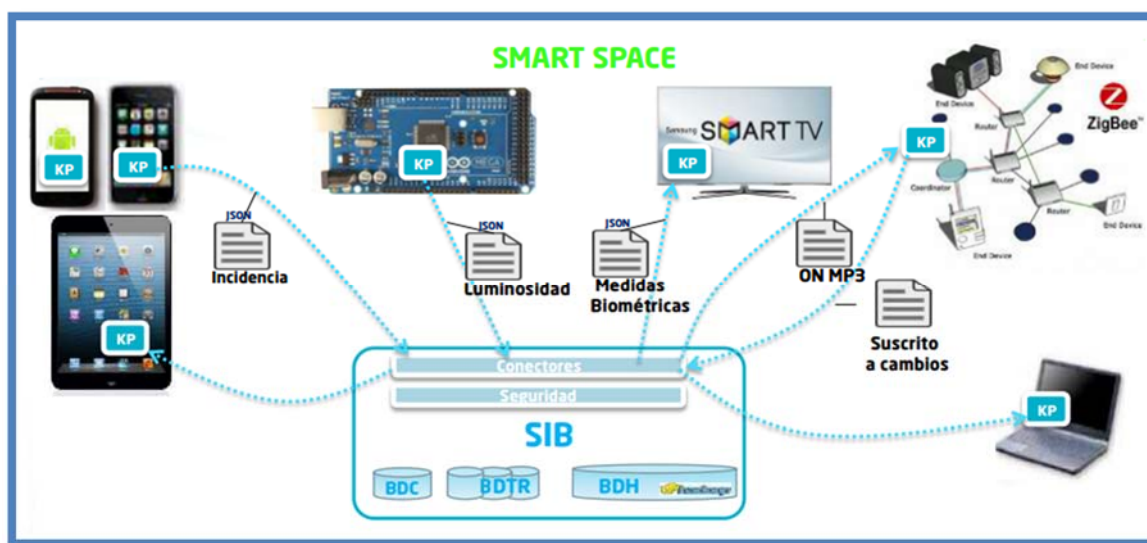
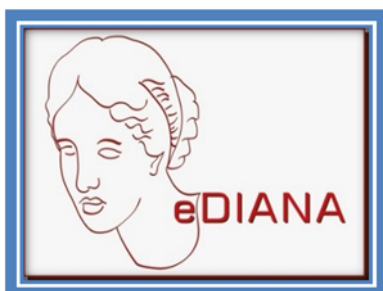


Fig. 1 Esquema nivel básico funcionamiento de Sofia2

6.2 eDiana Acciona



eDiana (Embedded Systems para Energy Efficient Buildings) responde a la necesidad de lograr la eficiencia energética en los edificios a través de soluciones innovadoras basadas en sistemas embebidos o empotrados.

El objetivo principal de eDiana, es permitir que la vida urbana sostenible a través de la racionalización en el uso de los recursos al tiempo que aumenta el confort en las zonas urbanas por medio de tecnologías de inteligencia e integración embebidos. El enfoque eDiana es lograr una mayor eficiencia en el uso de los recursos, dando prioridad a la energía como recurso escaso, más flexibilidad en la provisión de recursos y un mejor conocimiento de la situación para el ciudadano y para los propietarios de servicios e infraestructura. Esto se logrará a través de la implementación y la interoperabilidad de los sistemas integrados en todo el entorno de eDiana. El principal resultado de las aplicaciones de eDiana es en realidad la mejora de la eficiencia energética en los edificios residenciales y no residenciales.

Gracias a este proyecto, los edificios podrán convertirse en células totalmente activas en la distribución de la energía a través de la red eléctrica. En un futuro los edificios serán semejantes a organismos vivos, por lo que respecta a la interacción con respecto a la gestión, manejo y utilización de la energía. Con eDIANA estamos más cerca de la gestión inteligente de la energía, que podrá ser integrada en la domótica, siempre también, buscando la comodidad de los usuarios.

6.3 Libelium certified cloud partner



Es un dispositivo inalámbrico modular y horizontal que puede ser utilizado en cualquier escenario de la IoT (Internet of Things). Se conecta a Internet a través de wifi, Ethernet o 3G. Mediante una conexión a la nube, permite al usuario conectarse automáticamente a su plataforma.

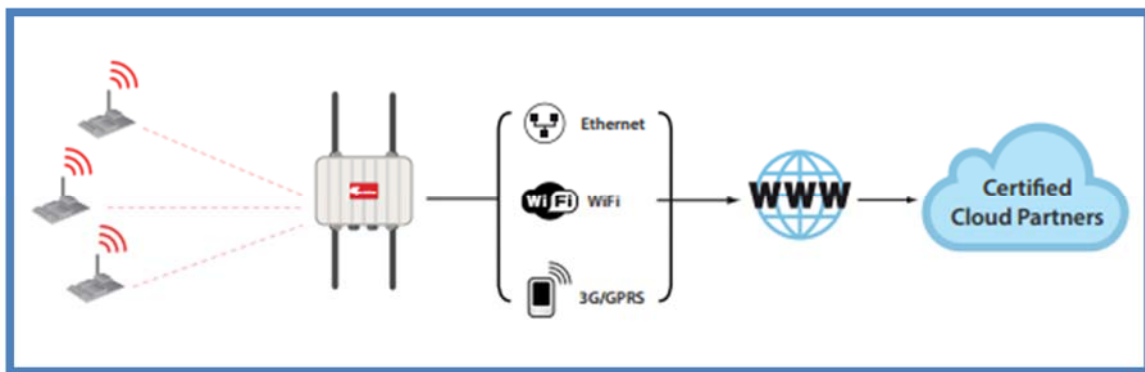


Fig. II Diagrama de funcionamiento de libelium certified cloud partner

El estado del Arte nos sirve para explorar que hay hecho y que con que cosas puedo contribuir con este proyecto, por lo que basándonos en esta idea de IoT y observando que estas empresas dejan un mercado sin explotar, que es donde quiero llegar con este proyecto, cogiendo esta idea y este funcionamiento, llevándolo a un nivel inferior pudiendo explotar este mercado donde el usuario básico (con limitaciones económicas), pueda disfrutar y utilizar el concepto IoT y las utilidades de estos dispositivos, mejorando el sistema de adquisición de datos de una manera práctica y sencilla.

7 IoT (Internet of Things)

7.1 Historia

En 1999, Kevin Ashton, un técnico británico que ayudó a fundar el Centro de Auto-ID en el Instituto de Tecnología de Massachusetts, acuñó el término "Internet de las cosas", pero la idea de conectar dispositivos entre sí, proviene de una fecha tan lejana como la creación del propio Internet. La cantidad de información que pueda ser creada, almacena y compartida va creciendo, al mismo tiempo, los lugares y las personas que antes parecían que se encontraban muy lejanas entre sí, ahora pueden conectarse e interactuar de una manera cada vez más cercana.

Los primeros sistemas con capacidad de conectarse a Internet no parecen ser nada novedosos a día de hoy, pero programadores e ingenieros desarrollaron el primer dispositivo conectado a Internet a principios de 1980, el dispositivo consistía en una máquina de Coca-Cola que enviaba actualizaciones y mensajes de estado acerca de la disponibilidad de una lata de Coca-Cola, para que un viaje a la zona de aperitivos no fuese en vano. Este término se basa en el concepto M2M (máquina a máquina) es, en síntesis, la capacidad de intercambiar datos entre dos máquinas remotas, de forma que mediante este intercambio, es posible controlar y supervisar de forma automática procesos en los que intervienen máquinas. El foco principal de aplicación de M2M se ubica por tanto, en los entornos relacionados con la telemetría y/o el telecontrol.

No fue hasta finales de 1990 y principios de 2000 que el concepto de tener una red de dispositivos interconectados se hizo popular y atrajo el interés de las empresas y los consumidores. Kevin Ashton lideró el movimiento en su Auto-ID Center del MIT con la investigación en el campo de la identificación por radiofrecuencia.

7.2 FI-WARE

Actualmente, el mayor problema al que se enfrenta el IoT es la falta de normas para la comunicación. Sin un "método de comunicación común", los dispositivos sólo podrán hablar con sus propias marcas y limitar severamente la utilidad de las máquinas conectadas. Por ello Telefónica implementó, como estrategia, la arquitectura llamada FI-WARE, el programa plantea el desarrollo de una plataforma realmente abierta para el desarrollo de las aplicaciones en la Internet del Futuro (FI-WARE) y la creación de un ecosistema de innovación abierto. El compromiso de Telefónica con el programa es absoluto, y Telefónica I+D lidera el desarrollo de la plataforma FI-WARE.

FI-WARE proporciona capacidades Cloud avanzadas basadas en el estándar abierto OpenStack sobre las que adicionalmente se ofrece una rica librería que facilitan el desarrollo de aplicaciones en múltiples sectores. La conexión con el "Internet de las Cosas", el almacenamiento, acceso, procesado, publicación y análisis tanto de contenidos multimedia como de datos a gran escala. Las especificaciones de la plataforma FI-WARE son públicas y libres, facilitando la aparición de múltiples proveedores y evitando que los emprendedores queden atados a un proveedor concreto.

En relación con la creación de un ecosistema sostenible de innovación abierto, buscando un método que destruya las barreras que nos encontramos en todo el ámbito de IoT, consolidando una única arquitectura para que sirva de base para cualquier plataforma. [1]

8 Arduino

8.1 Historia



La historia de Arduino comienza en Italia, en un instituto dedicado a la enseñanza de diseño interactivo en la ciudad de Ivrea, donde Massimo Banzi, uno de sus docentes, se propuso en 2003 diseñar su propia placa de hardware para trabajar con sus estudiantes. Parece que estudiantes y profesores entusiastas, combinados con el desafío de saltarse las barreras económicas que dificultan el acceso, son una fórmula ideal para propiciar cambios: quince años antes, Linus Torvals, en Finlandia, comenzó a diseñar su propio Kernel, basado en el código compartido por uno de sus profesores (Minix) solo porque las licencias UNIX estaban fuera del alcance de un estudiante común. Pero centrándonos en Arduino, los fundadores del proyecto, Massimo Banzi y David Cuartielles, junto con otros colaboradores, decidieron no llevar adelante su proyecto en soledad y publicaron los avances del mismo en la red, liberándolo como "software libre" y "hardware libre" (Los diseños del hardware se liberan con una licencia libre, de todas formas la definición de "Free Hardware" u "Open Hardware" aún son terreno de controversia, las leyes sobre propiedad intelectual en el mundo del hardware, no actúan de la misma forma que en el software u obras inmateriales). También se inclinaron por los microcontroladores de Atmel, que si bien son el elemento de hardware no-libre dentro de la placa, el fabricante publica sus librerías de compilación libres, de tal forma que resulta fácil para la comunidad portarlas a diferentes sistemas operativos.

Gracias a la colaboración de más interesados a través de la red, los prototipos evolucionaron hacia modelos más accesibles. En 2005 se comenzaron a fabricar las primeras placas y las bautizaron con el mismo nombre que uno de los hijos pródigos de la ciudad, "Arduino I, Marqués de Ivrea". El marqués llegó a rey de Italia en 1002. Y hoy hay fabricadas más de 120.000 placas Arduino por todo el mundo. Y el fenómeno sigue en aumento.

8.2 ¿Qué es y cómo funciona?

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinarios. Es un dispositivo con el cual conectamos el mundo analógico con el digital.

Está compuesto con un microcontrolador Atmel AVR con puertos entrada/salida, (I/O) de formato digital y analógico. Dependiendo del Arduino encontramos más o menos puertos y diferentes tipos de microcontroladores.

8.3 Puntos a favor de Arduino:

A la hora de elegir una placa de programación escogí Arduino por su bajo coste y mantenimiento, al mismo tiempo, al tener los microcontroladores Atmel y sus librerías de compilación libres suelen ser sencillos y permitiéndonos realizar múltiples funciones.

Otro gran punto a favor, que por el cual disminuye el coste total, disponemos de un software libre que consiste en un entorno de desarrollo que nos implementa un lenguaje de programación y la opción de cargar el lenguaje programado en la placa.

Al poder disponer de un software libre y la gran utilización de Arduino mundialmente, y no solo específicamente a nivel industrial, sino a nivel de usuario, nos proporciona una libertad y acceso a infinidad de información y librerías.

8.4 Puntos en contra de Arduino:

Al encontrarnos con un software libre, encontramos un entorno de desarrollo demasiado simple y con una gran complicación a la hora de corregir errores. El software no dispone de una simulación del código, como podemos observar en otros programas como KeilU.

Al trabajar con una placa de programación, ya sea Arduino u otra diferente, tendremos zonas limitadas de funcionamiento, respecto al entorno donde queremos utilizar nuestro sistema, ya que no tendremos una protección a nivel industrial como con un PLC.

El problema más grave que podemos encontrarnos en Arduino son fallos en algunas librerías, no especificando que nos funcione mal un programa creado, sino que sentencias que teóricamente son correctas, las librerías desprecian líneas de código, por lo cual encontramos momentos a la hora de programar en los cuales Arduino puede llegar a ser muy limitado.

9 Herramientas utilizadas

9.1 Hardware

Son todas aquellas partes físicas de un sistema electrónico, compuesta de componentes, eléctricos, electrónicos, electromecánicos y mecánicos.

9.1.1 Arduino MEGA 2560

Para la realización del TFG el Hardware principal es el Arduino MEGA2560.

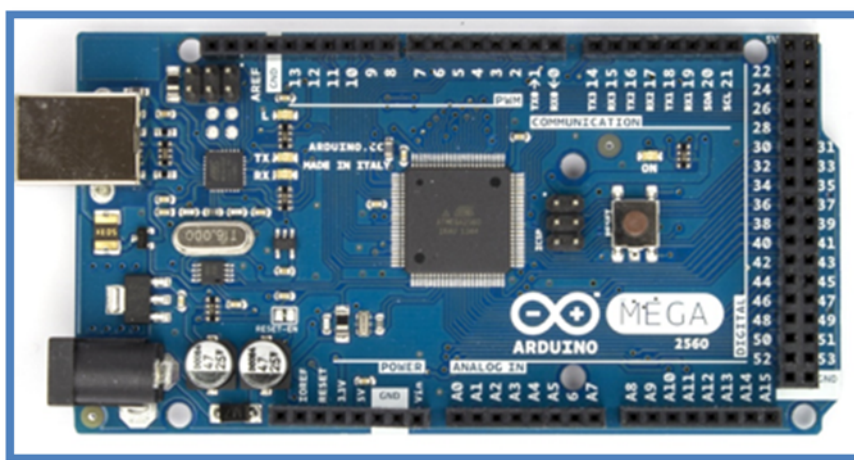


Fig. III Arduino Mega 2560

El Arduino Mega 2560 es una placa electrónica basada en el Atmega2560 (Datasheet [3]). Cuenta con 54 pines digitales de I/O, 16 entradas analógicas, 4 UART. ArduinoMega es compatible con la mayoría de los escudos diseñados para el Arduino y necesario para este proyecto al implementar un escudo Ethernet.

9.1.1.1 Características:

características	DESCRIPCIÓN
Microcontrolador	Atmega2560
voltaje operativo	3.3V-5V
alimentación(recomendada)	6-20V (7-12V)

características	DESCRIPCIÓN
Pines digitales I/O	54(de los cuales 15 PWM)
entradas analógicas	16 con 10 bit de resolución
Corriente Continua pines I/O	50mA(3.3V) / 40mA(5V)
Memoria Flash	256KB(de los cuales, 8 para bootloader)
Sram	8KB
EEPROM	4KB
Velocidad de reloj	16MHz

9.1.2 Entradas/Salidas (I/O)

El Arduino Mega 2560 proporciona 54 pines digitales que pueden ser usados como entrada o salida, usando las funciones `pinMode()`, `digitalWrite()` y `digitalRead()`. Funcionan a 5V y pueden proporcionar o recibir un máximo de 40 mA. Cada pin tiene una resistencia de pull-up interna de 20-50 kOhmios. Además algunos pines están reservados para alguna función en particular:

Serial: 0 (RX) y 1 (TX). Serial 1: 19 (RX) y 18 (TX); Serial 2: 17 (RX) y 16 (TX); Serial 3: 15 (RX) y 14 (TX). Se utilizan para recibir (RX) y para transmitir (TX) TTL datos en serie.

Interrupciones externas: Encontramos hasta 6 interrupciones controladas por los siguientes pines digitales: 32 (interrupción 0), 3 (interrupción 1), 18 (Interrupción 5), 19 (interrupción 4), 20 (interrupción 3) y 21 (interrupción 2). Estos pines pueden configurarse para activar una interrupción en un valor bajo, un flanco ascendente o descendente, o un cambio en el valor.

PWM: Pines del 2 al 13 y del 44 al 46. Proporcionan salida PWM de 8 bits con la función definida en la librería de Arduino `analogWrite()`.

SPI: Pines 50 (MISO), 51 (MOSI), 52 (SCK) y 53 (SS), estos pines soportan la comunicación SPI. El protocolo de datos encargado para la comunicación entre uno o más dispositivos periféricos.

9.1.3 Arduino Ethernet Shield:

Para realizar la comunicación entre todos los dispositivos, necesitamos añadir un escudo a ArduinoMega, en nuestro caso al realizar la comunicación TCP/IP utilizaremos el Arduino Ethernet Shield (Datasheet [2]).

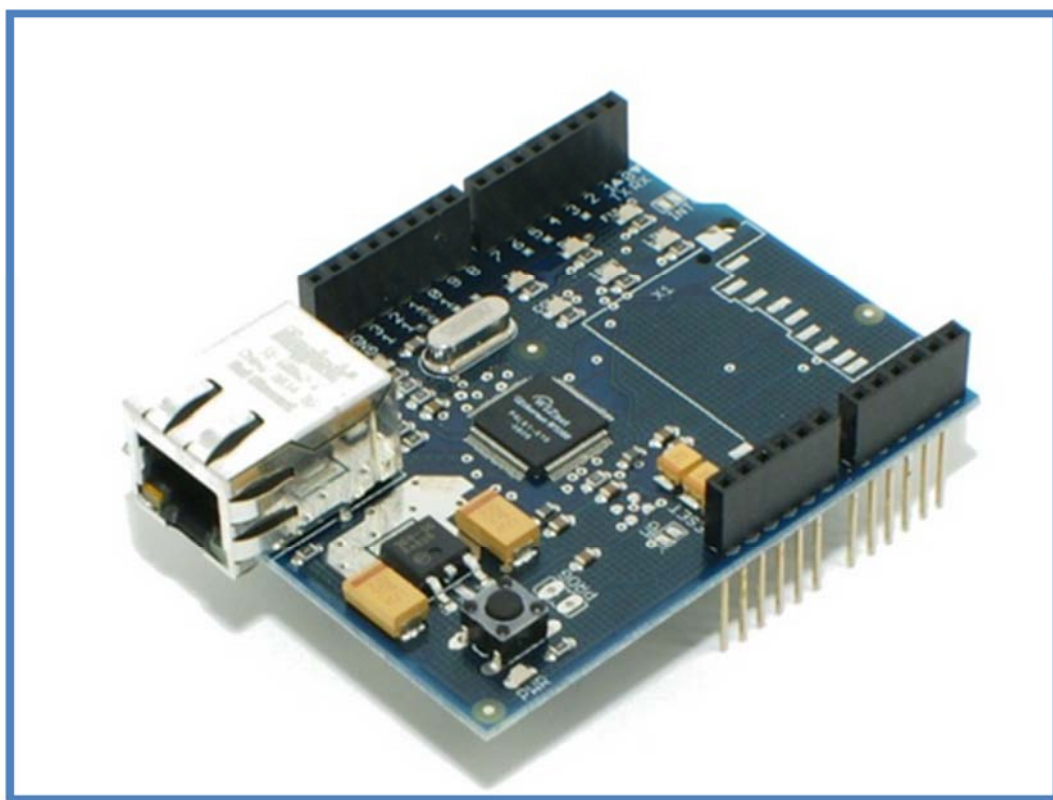


Fig. IV Arduino Ethernet Shield

Basada en el chip Ethernet Wiznet W5100, con el cual permite realizar una conexión TCP y UDP. Es capaz de realizar conexiones simultáneas.

9.1.3.1 Características y diagrama de bloque

- Protocolos soportados TCP / IP: TCP, UDP, ICMP, IPv4 ARP, IGMP, PPPoE, Ethernet.
- Ethernet integrado.
- Soporte MDI / MDIX.
- Soporte de conexión ADSL (con el apoyo Protocolo PPPoE con modo de autenticación PAP/ CHAP).

- Soporta 4 tomas independientes simultáneamente.
- No es compatible con la fragmentación IP.
- Memoria interna para 16Kbytes Tx / Rx.
- Tecnología CMOS de 0,18 μm .
- Operación 3.3V tolerando voltaje de E/S de 5V.
- Pequeña 80 Pin LQFP Paquete.
- Sin plomo Paquete.
- Interfaz periférico (SPI MODO 0, 3).
- Multifunción de las salidas LED (TX, RX, full / half duplex, collision, Link, Speed)

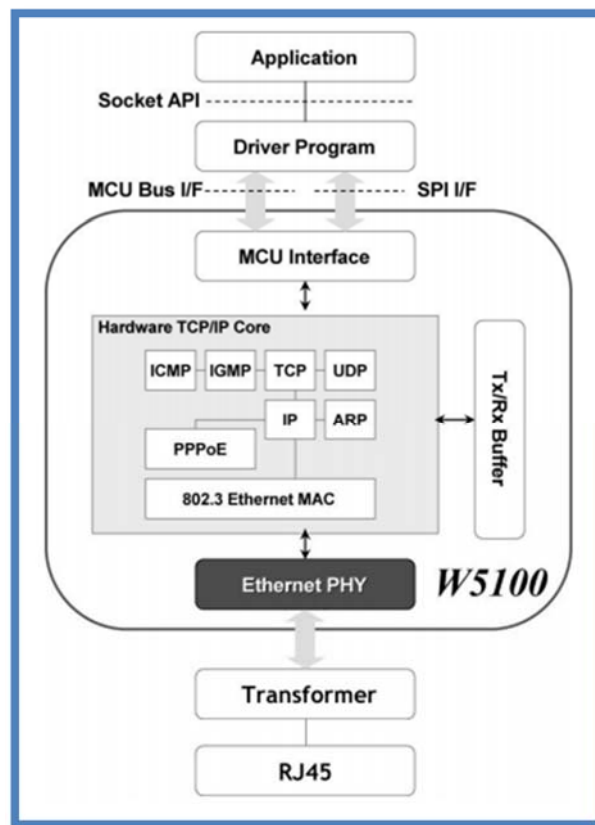


Fig. V Ethernet module

9.1.4 Sensor Waterproof DS18B20 Digital Temperature

Para el registro de temperaturas, el dispositivo a utilizar será el sensor DS18B20 resistente al agua, aparte de poder tener menor de restricción a la hora de trabajar a causa del entorno y su alto margen de temperaturas, no recibe

ningún tipo de degradación de la señal, realizando una comprobación podemos observar que a partir de los 20 m de distancia entre el sensor y el sistema de adquisición de datos obtenemos una pequeña degradación.



Fig. VI Sensor DS18B20

9.1.4.1 Características:

- Rango de temperatura: -55 a 125°C.
- Resolución: de 9 a 12 bits.
- Interfaz 1-Wire.
- Identificador interno único de 64 bits.
- Precisión: $\pm 0.5^{\circ}\text{C}$ (de -10°C a $+85^{\circ}\text{C}$).
- Tiempo de captura inferior a 750ms.
- Alimentación: 3.0V a 5.5V.

Debemos de destacar en este sensor su interfaz 1-Wire, siendo un protocolo de comunicación en serie, basándose en un bus con un maestro y varios esclavos alimentándose de una única línea de datos. Esto quiere decir que con este protocolo podemos conectar a un mismo pin de entrada digital todos los sensores que deseemos y mediante a nivel de código podremos detectar y trabajar con cada uno de ellos, al realizar ese montaje únicamente se necesita una única resistencia a la hora de alimentar el sensor, sin ser necesario

utilizar tantas resistencias como sensores, con estos sensores se facilita el montaje, y la utilización de escasas entradas digitales que nos quedarán libres para su utilización en otros dispositivos u objetivos

9.1.5 Buses de datos

9.1.5.1 Ethernet con conector RJ45



Fig. VII Cable Ethernet con conexión RJ45

El protocolo CSMA/CD incorpora dos mejoras, en primer lugar, nunca realiza una transmisión si hay otra estación hablando y en segundo si se encuentra transmitiendo y detecta otra estación que comienza hablar, produciéndose una colisión, detiene la transmisión, en lugar de seguir enviando la trama inútilmente.

Cada sistema puede enviar tramas de bits, uno a uno, colocando la trama en una misma conexión entre todos los sistemas, siendo capaces de ser vista por cada interfaz, pero cada trama envía unos bits de control donde se registra la dirección de destino.

9.1.5.2 USB



Fig. VIII USB 2.0 tipo A-B

Universal Serial Bus, un bus estándar industrial habitual de los ordenadores. Muchos fabricantes de instrumentos los integran facilitando el control del instrumento sin necesidad de adquirir hardware adicional, procediendo a una comunicación controlada y capaz de alimentar al periférico.

Al utilizar un USB 2.0 es definido como alta velocidad con una tasa de transferencia real máxima de 280Mbit/s. El cable dispone de cuatro líneas, dos de transmisión y recepción de datos, y los otros para alimentación del periférico.

La aparición del bus de datos USB fue a causa de que varias empresas quisieron unificar la forma de conectar periféricos sin necesidad de necesitar hardware específicos para la comunicación entre equipos de diferentes empresas

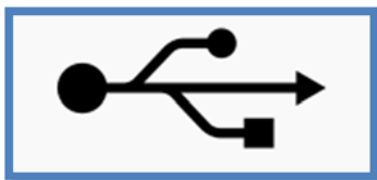


Fig. IX Símbolo USB

9.2 SOFTWARE

Se conoce como software a todo el equipamiento lógico de un sistema informático destacando aplicaciones informáticas como el procesador de texto, sistemas operativos... proporcionan al usuario un interfaz. El software complementa al hardware para realizar tareas específicas.

9.2.1 C++/C#

El lenguaje de programación utilizado para la creación de los códigos de Arduino servidor y Arduino cliente se basa en C++, que extiende el lenguaje C permitiendo la manipulación de objetos y estructuras, añadiendo facilidades de programación genérica definiendo de este modo al lenguaje C++ como un modelo de programación multiparadigma. Para completar la definición se declara un paradigma como una propuesta tecnológica validada por una comunidad de programadores siendo incuestionable.

9.2.2 Arduino 1.0.5-r2

El entorno de desarrollo de Arduino se compone de un editor de texto para poder escribir el código, un área de mensajes, una consola de texto, una barra de herramientas... Este entorno de desarrollo se conecta al hardware de Arduino para cargar programas y mantener una comunicación serie.

Podemos gestionar diferentes documentos con diferente extensión, archivos sin extensión, archivos C# (.c), archivos C++ (.cpp) o archivos de cabecera (.h).

A destacar en el entorno de Arduino, utilizamos librerías especiales, para poder controlar varios aspectos del programa:

Librería OneWire & DallasTemperature: Con esta librería realizamos la lectura recogida por los sensores térmicos digitales DS18B20.

Librería EEPROM: Utilizada para poder almacenar datos en la EEPROM de la placa y de este modo conservarlos de manera permanente.

Librería SPI: Librería necesaria para el interfaz periférico en serie.

Librería Ethernet: Un archivo cabecera encargado de la comunicación entre dispositivos, creando socket, y realizando hincapié en esta librería ya que se debe realizar una modificación (Problemas y soluciones).

9.2.3 LabView 2012

LabView (Laboratory Virtual Instrumentation Engineering Workbench) es una plataforma para diseñar sistemas con un lenguaje gráfico (G-Code). Esta plataforma nos permitirá realizar el SCADA del sistema, siendo el software creado para supervisar en nuestro caso los dispositivos de campo (sensores de temperatura), y de este modo se recogen los datos adquiridos de manera automática. Los programas creados por LabView se denominan como instrumentos virtuales (VIs).

El principal aspecto positivo del LabView es su fácil uso y programación, llegando a una utilización a nivel profesional como a nivel de usuario, al utilizar un lenguaje gráfico nos permite una programación menos compleja y en algunos

casos más intuitiva. Se tiene acceso a una gran cantidad de librerías con las cuales podemos aumentar el uso del LabView, todas ellas proporcionadas por el creador de LabView National Instruments. Presenta una gran facilidad de manejo en los siguientes aspectos:

1.-Interfaces de comunicaciones:

- *Puerto serie/paralelo*
- *GPIB*
- *PXI*
- *VXI*
- *TCP/IP, UDP, DataSocket*
- *Irda*
- *Bluetooth*
- *USB*
- *OPC*
- *FTP*
- *HTTP client*

2.-Herramientas gráficas y textuales para el procesamiento digital de señales.

3.-Visualización y manejo de gráficas con datos dinámicos.

4.-Adquisición y tratamiento de imágenes.

5.-Capacidad de interactuar con otros lenguajes y aplicaciones:

- *DLL: librerías de funciones*
- *.NET*
- *ActiveX*
- *Multisim*
- *Matlab/Simulink*
- *AutoCAD, SolidWorks, etc.*

6.-Control de movimiento (combinado incluso con todo lo anterior).

7.-Tiempo Real estrictamente hablando.

8.-Programación de FPGAs para control o validación.

9.-Sincronización entre dispositivos.

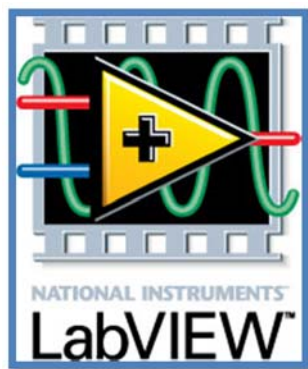


Fig. X Logotipo LabView

10 Diseño realizado

El diseño realizado consiste en la comunicación entre un Arduino Maestro con varios Arduinos Clientes para la adquisición de datos, en nuestro caso la adquisición de datos corresponderá a temperaturas registradas mediante un sensor digital de temperatura y de este modo monitorizar las magnitudes en grados de varias plantas farmacéuticas.

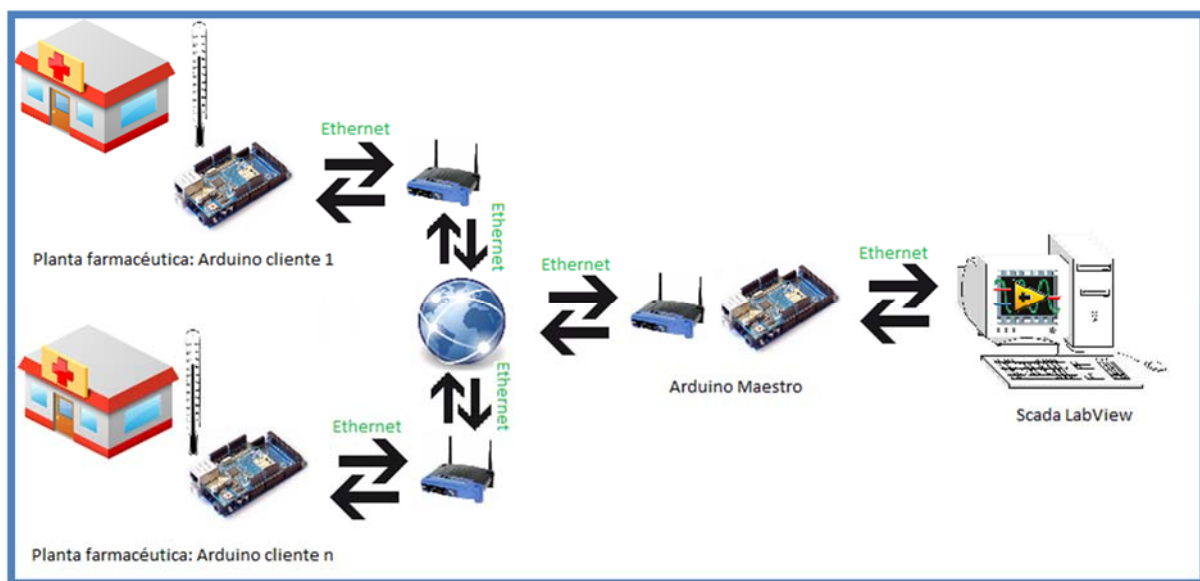


Fig. XI Esquema diseño realizado

En el esquema del diseño realizado podemos observar las conexiones y diferenciamos en tres partes, una parte de adquisición de datos, una segunda de registro de los mismos y la última parte la monitorización de todos ellos para su visualización. La conexión entre dispositivos a Internet es mediante Ethernet, es decir, la conexión entre los Arduinos y los routers es mediante cable (Ethernet) representado en nuestro esquema como dos flechas, simbolizando conexiones bidireccionales. Utilizamos un switch para crear una red personal pero el fin del sistema es la instalación y conexión directa a internet y no a una red local, aunque también podría utilizarse para un entorno cerrado.

10.1 Base teórica

El proyecto está enfocado en el territorio de IoT (Internet of Things) realizando una conexión de las cosas cotidianas a Internet, y no únicamente utilizando la conexión para los usuarios, la base teórica se fundamenta en esta idea (IoT), donde la finalidad de este proyecto es la conexión de sensores de temperatura a Internet pudiendo ser gestionados por otros equipos.

10.1.1 Telemedida

La telemedida es una aplicación de telecomunicaciones, que se basa principalmente en la transmisión de datos existiendo una cierta distancia entre emisor y receptor, la cual sea lo suficientemente grande para afectar al requerimiento de la estructura del sistema.

En este caso nos permite la recepción y registro de datos mediante Internet, pudiendo realizar una visualización y registro de los datos obtenidos.

10.1.2 Comunicación TCP/IP:

La comunicación entre dispositivos es mediante **TCP/IP** que consiste en identificar el grupo de protocolos haciendo posible la transferencia de datos entre sistemas, se destacan dos protocolos por separado, IP o protocolo de Internet y el TCP o protocolo de control de transmisión, las ventajas de esta comunicación es la fiabilidad siendo el método más utilizado para sistemas grandes o medianos, y el método de conexión a Internet, pero es una comunicación lenta si se encuentra en redes de un volumen de tráfico medio-bajo, a diferencia del modelo UDP que es más rápido pero menos fiable pudiendo perder paquetes en la transmisión de los datos, pero más sencillo de configurar, mientras que el TCP/IP es más complejo de configurar y de mantener. El sistema TCP/IP es el modelo real o aplicación del modelo OSI de la red, siendo este el modelo de referencia.

Dentro del modelo TCP/IP encontramos cuatro capas:

1.-Capa de aplicación: Incorpora aplicaciones de red estándar, siendo el más cercano al usuario.

2.-Capa de transporte: Enrutamiento de datos y mecanismos para conocer el estado de transmisión.

3.-Capa de Internet: Proporciona el paquete de datos.

4.-Capa de acceso a la red: Define el proceso de enrutamiento de los datos.

En la Fig. XII apreciamos que pasos sigue la comunicación entre dos sistemas enlazados por el sistema TCP/IP.

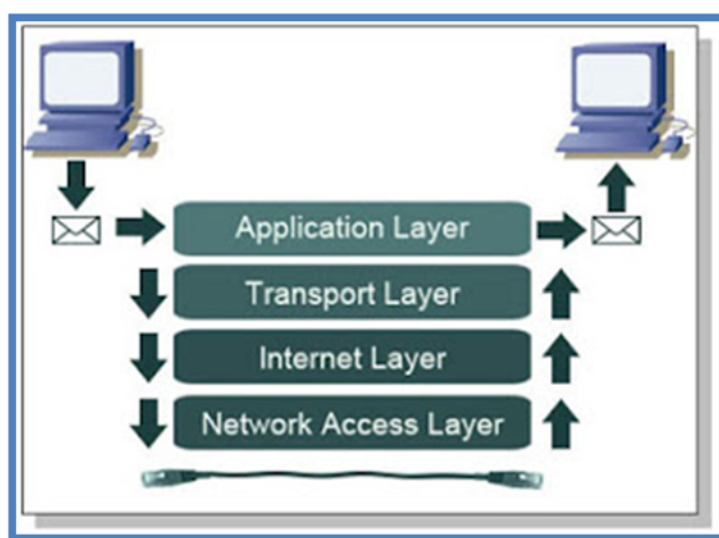


Fig. XII Capas TCP/IP

10.1.2.1 En el modelo OSI

En el modelo OSI encontramos siete capas:

1.-Capa de aplicación: Brinda aplicaciones al interfaz siendo el nivel más cercano al usuario.

2.-Capa de presentación: Define los datos (cifrado), los cuales serán manejados por la primera capa.

3.-Capa de sesión: Define inicio y fin de la comunicación entre sistemas.

4.-Capa de transporte: Realiza el transporte de datos, separación en paquetes y controlar errores de transmisión.

5.-Capa de red: Administra las direcciones y enrutamiento los datos.

6.-Capa de enlace de datos: Define la interfaz mediante la tarjeta de red.

7.-Capa física: Define el modelo de transformación de datos físicos en señales digitales.

En la Fig. XIII Relación OSI-TCP/IP" se aprecia la relación entre ellas por colores de las capas de cada modelo.

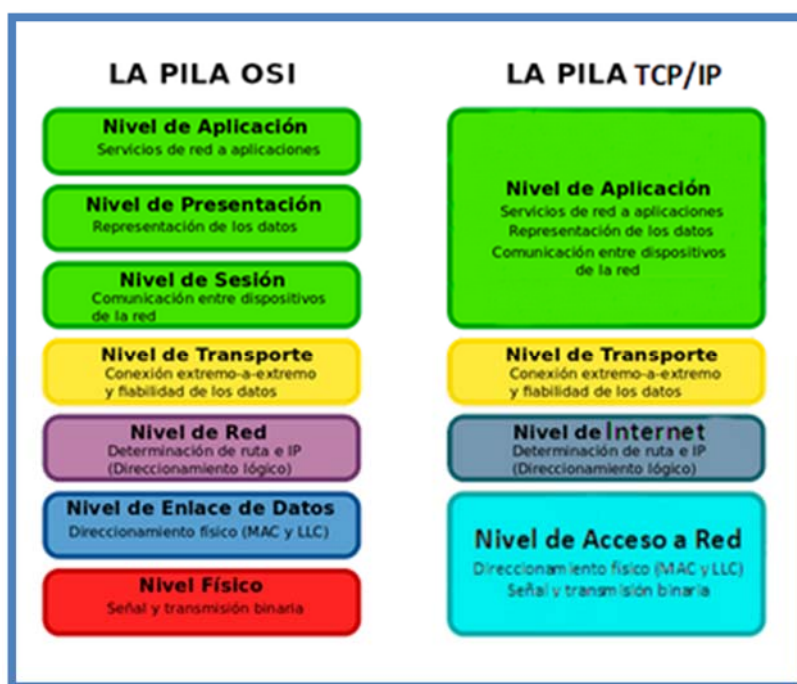


Fig. XIII Relación OSI-TCP/IP

10.1.2.2 Comunicación TCP/IP entre Arduinos

Inicialmente para poder tener la opción de conectar nuestro Fig. III Arduino Mega 2560 a Internet es necesario implementarle un Fig. IV Arduino Ethernet Shield, utilizando el protocolo anteriormente citado (Comunicación TCP/IP:) se basa principalmente en una comunicación cliente-servidor, es decir, los nodos utilizados, en este caso dos, trabajan como clientes que demandan atención al servidor o maestro, para nosotros otro Arduino. Inicialmente se realizó una conexión entre los nodos y un SCADA creado mediante LabView 2012, sin disponer de un Arduino maestro funcionando como Gateway entre los nodos y nuestro SCADA. La librería principal utilizada para nuestra conexión entre el Arduino cliente y el Arduino server es la librería entandar de Ethernet,

procediendo a la conexión entre ambos dispositivos mediante las siguientes sentencias:

- **begin():** Le dice al servidor que comience a escuchar las conexiones entrantes. Sintaxis: `server.begin()`.
- **connect:** Se conecta a una dirección IP y un puerto. El valor de retorno indica el éxito o el fracaso. Sintaxis: `client.connect(ip, port)`,
- **available() cliente:** Devuelve el número de bytes disponibles para la lectura (es decir, la cantidad de datos que ha escrito el cliente por el servidor que está conectado). Sintaxis: `client.available()`
- **available() server:** Obtiene un cliente que está conectado al servidor y tiene datos disponibles para su lectura. Sintaxis: `server.available()`
- **Ethernet.begin():** Inicializa la librería Ethernet UDP y la configuración de red. Sintaxis `Ethernet.begin(puerto)`.
- **println():** Imprime números como una secuencia de dígitos, siendo cada uno un carácter ASCII. Sintaxis: `client.println()`, `server.println()`.
- **read():** Leer el siguiente byte recibido en el servidor del cliente que está conectado después de la última llamada de leer (`read`). Sintaxis: `client.read()`
- **write() cliente:** Escribir datos en el servidor en el cual el cliente está conectado. Estos datos se envían como un byte o una serie de bytes. Sintaxis: `client.write()`
- **write() server:** Escribir datos en todos los clientes conectados a un servidor. Estos datos se envían como un byte o una serie de bytes. Sintaxis: `server.write()`.
- **flush():** Desecha todos los bytes que se han escrito para el cliente pero aún no leídos. Sintaxis: `client.flush()`
- **stop():** Finalizamos la conexión entre cliente-servidor. Sintaxis: `client.stop()`.

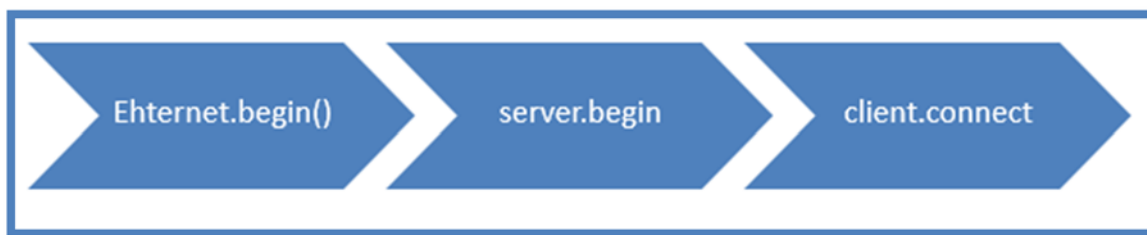


Fig. XIV Inicialización Servidor-Cliente

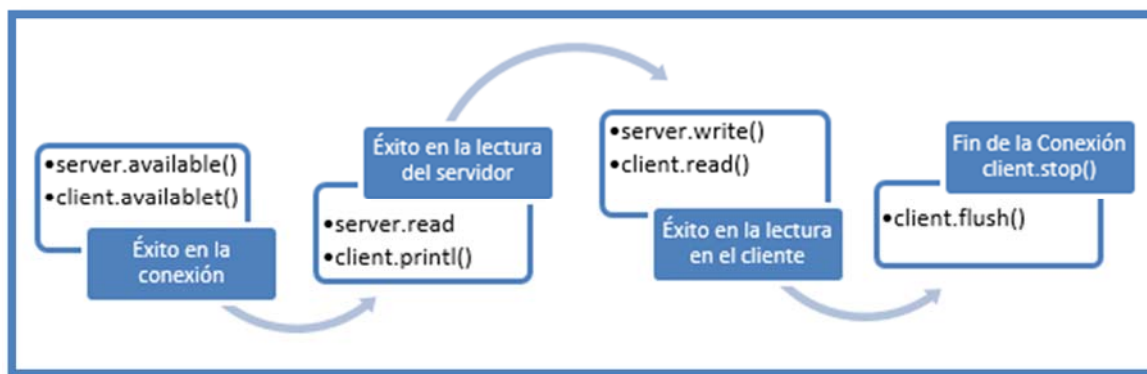


Fig. XV Comunicación servidor-cliente

Con estas sentencias definidas anteriormente realizamos la comunicación entre cliente y servidor, pero se han de definir los servidores y los clientes, y la dirección IP utilizada para nuestro sistema:

- **EthernetServer():** Crear un servidor que recibe las conexiones entrantes en el puerto especificado. Sintaxis: `server(puerto)`
- **EthernetClient():** Crea un cliente que puede conectarse a una dirección IP especificada Internet y el puerto (que se define en la función `client.connect ()`).
- **IPAddress():** Define una dirección IP. Se puede utilizar para declarar direcciones locales y remotas. Sintaxis: `IPAddress(dirección)`.

Con todas estas funcionalidades somos capaces de crear un servidor y varios clientes realizando una comunicación entre ellos. Aparte de la librería Ethernet, se necesita la librería SPI.h para realizar cálculos y sentencias para el manejo de los datos obtenidos en cada dispositivo.

10.1.3 Adquisición de datos

Para realizar la adquisición de datos con el sensor de temperatura Sensor DS18B20 (Fig. VI Sensor DS18B20) el cual proporciona lecturas en centígrados con una precisión de 9 a 12 bits, y un rango de trabajo comprendido entre -55°C y 125°C, basándose en un interfaz One-Wire.

10.1.3.1 ¿Qué es ONE-WIRE?

One-Wire es en realidad un sistema simple de dispositivos tales como dispositivos de vigilancia meteorológica, dispositivos de monitoreo de jardín, o dispositivos de automatización del hogar que están conectados a través de una

red One-Wire a un dispositivo, microcontrolador, ordenador, PLC... El dispositivo recibe los datos de los sensores, lo que permite realizar un seguimiento de las precipitaciones, la temperatura, la humedad, los niveles de humedad del suelo...

10.1.3.2 ¿Cómo funciona?

One-Wire consta de un controlador maestro que está conectado a uno o varios dispositivos esclavos. El controlador maestro es típicamente una computadora o un microcontrolador con una interfaz One-Wire externo. Todos los sensores son los dispositivos esclavos. El maestro se comunica con uno o más dispositivos esclavos usando el protocolo serial One-Wire desarrollado por Dallas, envío y recepción de señales a través de **una sola línea de datos** además de referencia de tierra. El protocolo One-Wire sincroniza los dispositivos esclavos al maestro.

Una característica clave del sistema de Dallas es que cada dispositivo esclavo One-Wire tiene una dirección única (MAC), de este modo el maestro podrá identificar individualmente a los esclavos aunque la información se introduzca por un mismo pin de entrada de nuestro Arduino (en nuestro caso).

Para realizar las mediciones con Arduino necesitamos utilizar las librerías específicas para obtener los datos mediante la tecnología One-Wire, son las librerías de DallasTemperature.h y OneWire.h, con las siguientes sentencias necesarias para su correcto funcionamiento:

- **OneWire:** Define una instancia One-Wire para comunicarnos con cualquier dispositivo One-Wire. Sintaxis: `OneWire onewire()`.
- **DallasTemperature:** Se debe pasar la referencia a la librería de Dallas Temperature. Sintaxis: `DallasTemperature sensors(&onewire)`.
- **begin():** Inicia la recepción de One-Wire. Sintaxis: `sensors.begin()`.
- **getTempCByIndex():** Realiza la lectura de un sensor. Sintaxis: `sensors.getTempCByIndex()`.

10.1.4 SCADA LabView

SCADA (Supervisory Control And Data Acquisition) es un software el cual nos permitirá visualizar y controlar el procedimiento de la adquisición de datos,

con el cual podremos visualizar en pantalla los resultados obtenidos y almacenarlos, el SCADA de esta aplicación estará basado en LabView.

Para realizar una comunicación TCP/IP debe existir dos sistemas que se comuniquen entre sí, una comunicación pasiva y otra activa, siendo la primera la que espera por una conexión entrante, para nuestro proyecto nuestro programa debe de ser un sistema activo, llamando al Arduino maestro, teóricamente tendremos dos pasos importantes, el primero será crear la conexión entre dispositivos (entre el Arduino y LabView) como podemos observar en la Fig. XVI Diagrama de flujo del establecimiento de conexión en el sistema activo.

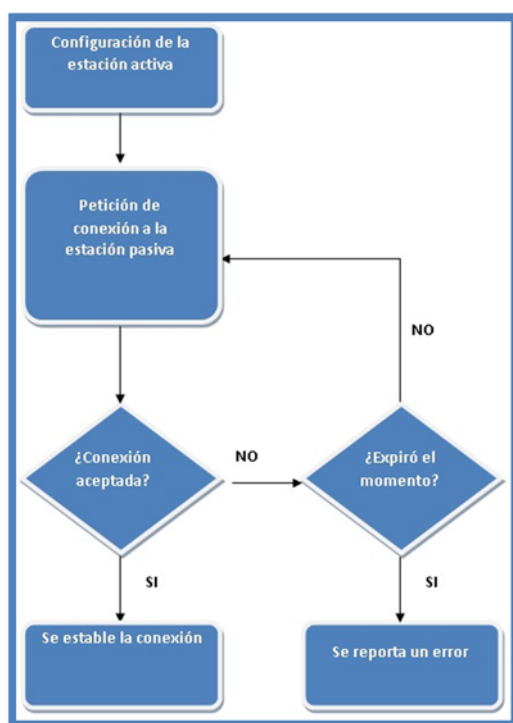


Fig. XVI Diagrama de flujo del establecimiento de conexión en el sistema activo.

Como podemos observar en el flujo de establecimiento de la conexión, debemos realizar la configuración, la estación pasiva espera el tiempo establecido a que otra estación inicie una comunicación por el puerto programado, la estación activa llama a la estación pasiva y solicita la conexión por un puerto específico, si la estación pasiva detecta que una conexión se está solicitando a través del puerto establecido se establece la comunicación y tanto la estación activa como pasiva queda en capacidad para enviar y recibir datos, en la Fig. XVI se pueden apreciar los diagramas de flujo correspondientes a estas

operaciones.

El segundo punto importante es la recepción y lectura de los datos enviados por el Arduino maestro, en la Fig. XVII Diagrama de flujo de una operación de lectura o recepción de datos TCP/IP se puede apreciar el diagrama de flujo correspondiente a esta acción.

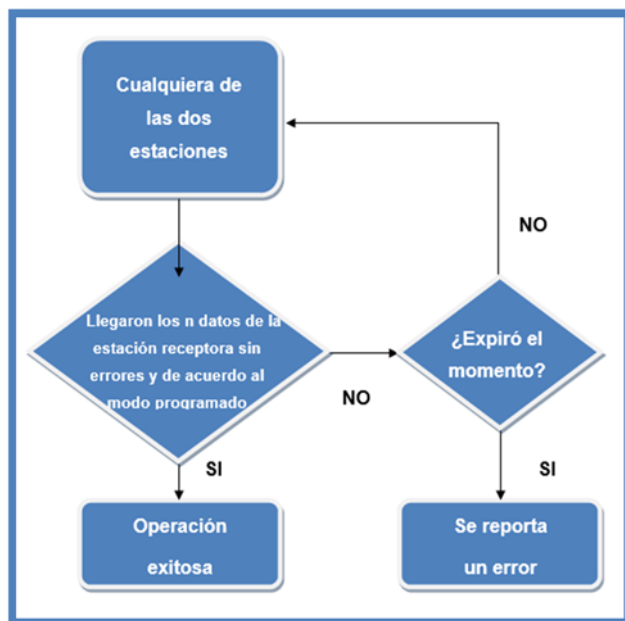


Fig. XVII Diagrama de flujo de una operación de lectura o recepción de datos TCP/IP

Cualquiera de las dos estaciones envía un número de datos determinado, si la estación receptora no recibe todos los datos y expira el tiempo reporta un error en la estación emisora, en la Fig. XVII se muestra el diagrama de flujo correspondiente, y como se puede apreciar, si durante la operación de lectura cualquiera de las dos estaciones que han establecido una conexión y la estación receptora espera a los datos enviados, y si recibe todos los datos, la operación será exitosa y no reportará ningún error.

10.2 Base práctica

Para la realización de este proyecto inicialmente seguimos varios pasos hasta llegar a nuestro objetivo final, pudiendo mostrar todos y cada uno de los problemas encontrados en cada paso

10.2.1 Pasos previos

10.2.1.1 Primer paso

El paso inicial fue crear la comunicación entre un único Arduino y un SCADA sencillo para visualizar cualquier dato enviado desde Arduino.

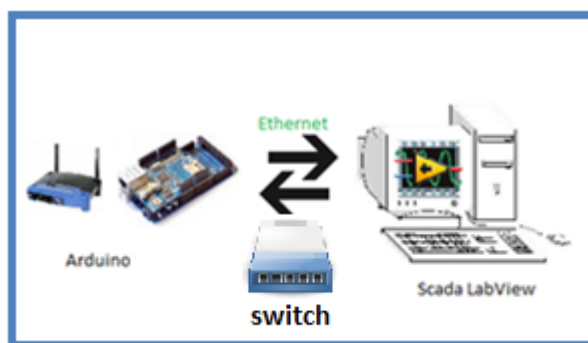


Fig. XVIII diseño inicial

Con este diseño inicial se pudo conseguir realizar una comunicación TCP/IP de manera sencilla y para la obtención de una única información, en este caso envía un valor determinado y mostraba por pantalla. Para realizar este procedimiento utilizando un switch configurando nuestra IP.

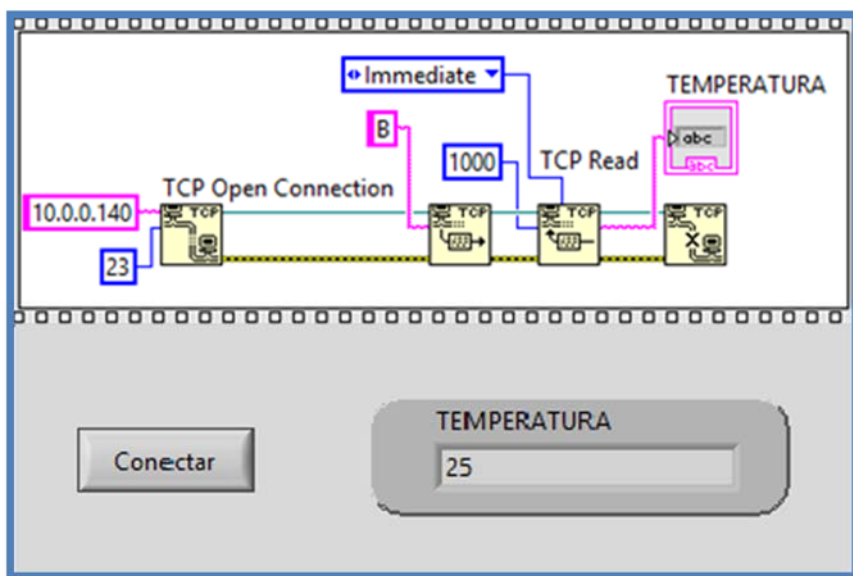


Fig. XIX Resultado diseño inicial

Como observamos en la Fig. XIX, la parte superior corresponde al diagrama de bloques, donde realizamos la operación de conexión entre dispositivos, seguidamente se realiza una escritura que será enviada al Arduino (en este caso B), una vez realizada esta escritura, el Arduino comenzará a enviar un dato y nuestro sistema realizará la lectura mediante el bloque “TCP Read”, y dicho valor se mostrará por pantalla (parte inferior de la figura correspondiente al panel frontal), por último se cierra la conexión entre dispositivos pudiendo realizar de nuevo otra lectura activando el botón “Conectar”.

Podemos destacar en este paso que la lectura realiza por el programa creado en LabView, siempre nos devuelve un string (cadena de caracteres), que para mostrar un entero no nos muestra ningún error, pero nos encontraremos con un problema en el momento que en vez de enviar un “25”, recojamos una temperatura cuantificada y registremos un valor con parte decimal.

El problema encontrado en este paso fue que si nuestro Arduino comenzaba la conexión con el SCADA generado y no conseguía realizarla de manera satisfactoria el propio SCADA nos devolvía un error, quedando el programa en standby y se debía realizar un reinicio de manera continuada, el problema se solucionó en un primer momento introduciendo un periodo de tiempo grande en el SCADA para intentar realizar la conexión, no siendo una corrección adecuada ni recomendable, ya que si en un momento el Arduino dejase de funcionar, el programa realizado con LabView daría el mismo problema. Finalmente se solucionó introduciendo una estructura “while” como podemos observar en la siguiente figura:

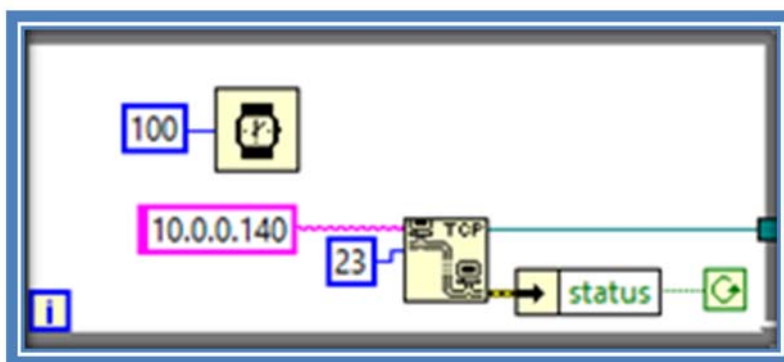


Fig. XX estructura while

10.2.1.2 Segundo paso

Al poder realizar la comunicación con un único Arduino y cerciorarnos que realizaba una lectura correcta, nos centramos en el siguiente objetivo, que era realizar una obtención de un dato enviado desde un Arduino con otro como “puente”, es decir, el segundo Arduino trabajando como una puerta de acceso (Gateway). En este punto la configuración con LabView era exactamente la misma, y únicamente se debía realizar la configuración del Arduino cliente con el Arduino Maestro, definiendo en el código del mismo la dirección IP a la cual se conectará y un puerto asignado de conexión con el Arduino que trabajara como “puente” (Arduino Maestro).

Inicialmente encontramos problemas realmente graves a la hora de crear el socket a causa de la librería Ethernet, donde un bug (error de software) produce un fallo al crear la conexión, sin darnos ningún error en el código ya que inicialmente realizaba la sentencia de manera satisfactoria pero sin conseguir el fin requerido para nuestro sistema. El fin es que el Arduino cliente nada más realizar la conexión con el maestro, comience a enviar datos, pero si el maestro no escribe absolutamente nada en dirección al cliente, este cierra la conexión sin llegar a enviar ningún dato, solucionamos este error modificando la librería Ethernet (pág. 77 Comunicación Arduino cliente y maestro).

10.2.1.3 Tercer paso

Una vez llegado al punto de poder recibir datos de un Arduino cliente, nuestro siguiente objetivo es la conexión de tres Arduinos clientes distintos, después de realizar el paso anterior es sencillo, únicamente a la hora de conectarse cada cliente al Arduino maestro, se definió a cada uno un puerto distinto al anterior. En la Fig. XXI Conexión tres clientesse demuestra que cada valor adjudicado a cada Arduino era leído de manera satisfactoria por LabView, en la Fig. XXI el primer valor es el adquirido por el Arduino maestro, ya que aparte de trabajar como nodo principal del sistema y recoger todos los datos externos, también trabajará de manera secundaria como otro sistema de adquisición de datos.

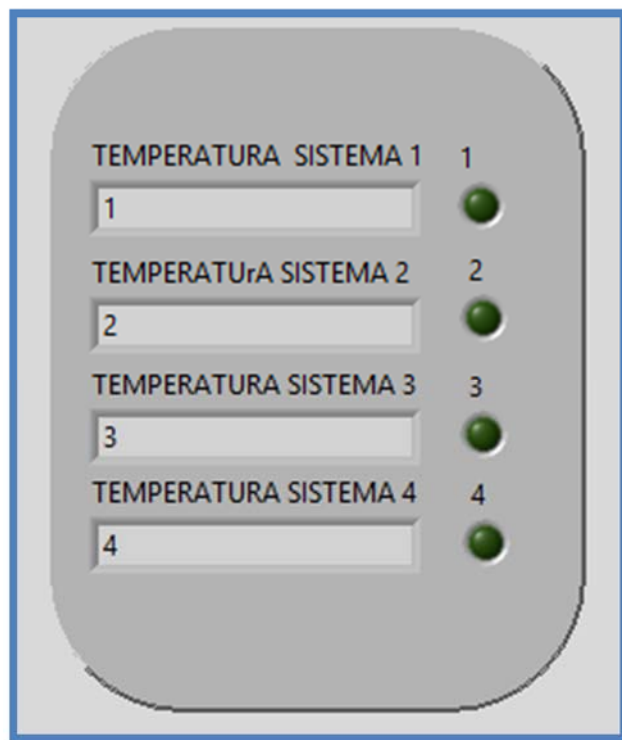


Fig. XXI Conexión tres clientes

Como mostramos en la imagen anterior, podemos observar la comunicación entre el Arduino maestro con los distintos clientes, obteniendo unos valores fijos enviados por cada Arduino (1, 2, 3 y 4).

```
Numero de clientes:  
1  
Client  
1  
connected.  
Client  
1  
disconnected.
```

Fig. XXII Comunicación entre Arduinos

En este paso observamos como para cada cliente realiza una conexión distinta, recibiendo de cada uno la temperatura enviada. El problema surgió a la enviar datos con parte decimal, donde podíamos observar que el dato registrado en el Arduino maestro era únicamente la parte entera, introduciendo dos ceros

en la parte decimal y de este modo el sistema quedaba muy limitado ya que únicamente podríamos registrar temperaturas con una resolución de un grado. La solución a seguir fue sencilla, desde el cliente realizamos dos escrituras, una que es la parte entera y otra que será la parte decimal, reconstruyendo el dato y consolidándolo en el Arduino maestro. A la hora de enviar el dato al SCADA no encontramos ningún error parecido.

10.2.1.4 Cuarto paso

La necesidad de realizar un sistema autosuficiente capaz de auto controlarse en algunos aspectos, como la creación de nuevos clientes y su registro en caliente, con el fin de no tener que reiniciar el sistema para incluir nuevos clientes, porque de este modo obtendremos un sistema más autónomo, para conseguir este propósito tuvimos que realizar la modificación del propio Arduino maestro siendo capaz de captar si aparecía un nuevo cliente y de este modo poder registrarlo y almacenarlo de tal modo que sea capaz el sistema de poder conectar un número de clientes determinados y ampliarlos si fuese necesario, y de este modo únicamente modificar el SCADA para un número elevado de clientes.

Para este proceso se necesita utilizar la EEPROM de los Arduinos, siendo la EEPROM un tipo de memoria no volátil, para poder registrar en cada sistema un valor y mantenerlo aunque se deje de alimentar el microcontrolador. Los pasos seguidos son sencillos, el maestro estará escuchando continuamente los puertos definidos como acceso para los clientes, con un puerto específico para los nuevos clientes (en nuestro caso el 100), de este modo siempre que conectemos un nuevo cliente con el puerto 100 definido, el Arduino maestro responderá a su petición de agregar un nuevo cliente, asignándole un nuevo puerto y el cliente registrando el valor de ese puerto en la EEPROM para que de este modo siempre que volvamos a conectar este mismo cliente, comience a comunicarse con el maestro enviando los datos requeridos. Por otro lado el maestro registrará que existe un nuevo cliente y guardará este mismo dato en su EEPROM para de este modo no escuchar indefinidos puertos, sino

únicamente los necesarios. En el siguiente diagrama de flujos (Fig. XXIII) podemos hacernos una idea como transcurre esta operación.

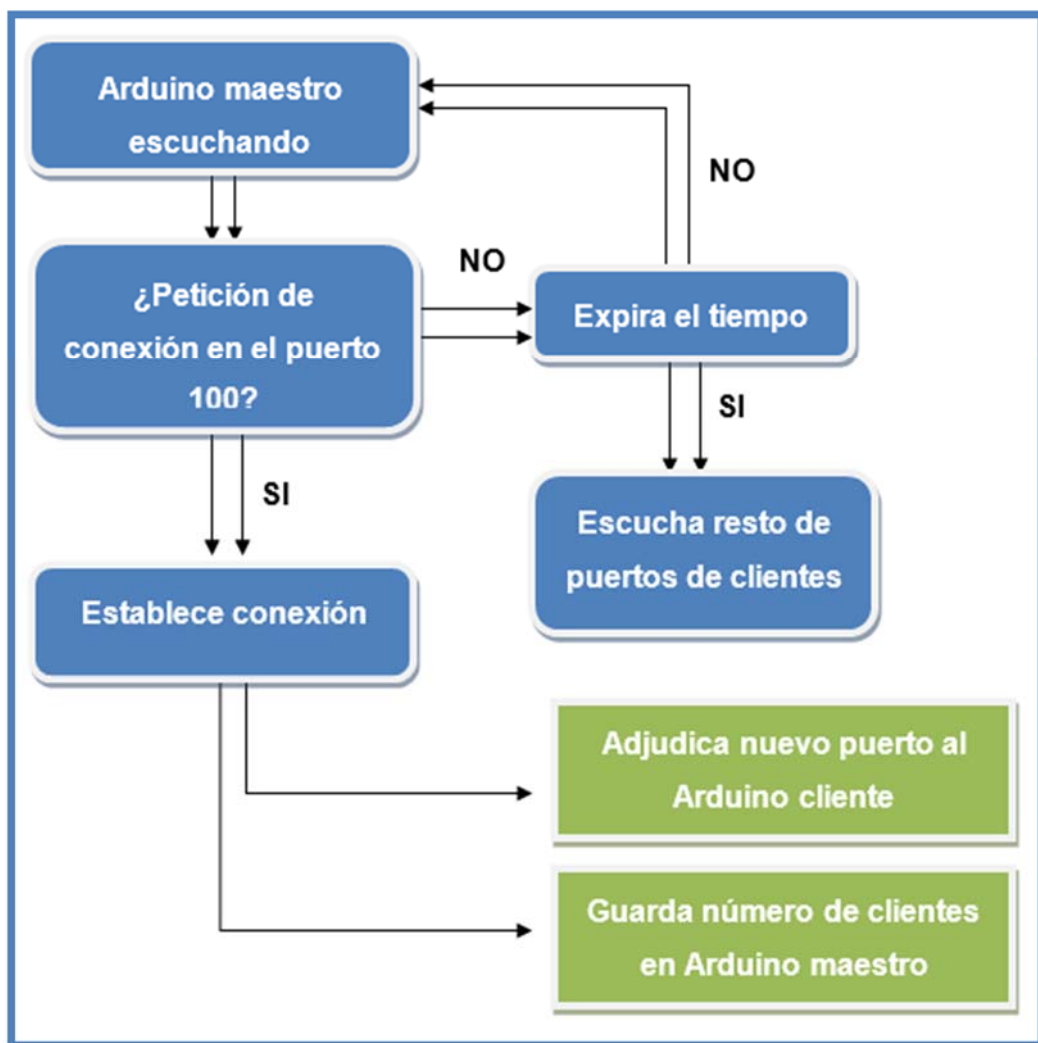


Fig. XXIII Diagrama de flujo nuevo cliente

Por otro lado incluimos mensajes como respuesta de la conexión y creación de un nuevo cliente en cada sistema, y de este modo monitorizar la comunicación entre ellos de forma gráfica obtenemos la siguiente respuestas de cada Arduino como se puede apreciar en las Fig. XXIV y Fig. XXV, mediante el puerto serie de cada Arduino.

```
Numero de clientes:
0
NUEVO CLIENTE
DeNumero de clientes:
1
Client
1
connected.
Client
1
disconnected.
```

Fig. XXIV Respuesta Arduino Maestro-nuevo cliente

```
connected
Parece ser que tenemos otro cliente...
100
***Nuevo puerto:***
101
101
disconnecting...
```

Fig. XXV Respuesta Arduino cliente-nuevo cliente

A partir de estos puntos realizáremos una descripción TOP-DOWN para una mejor apreciación del proyecto en su conjunto.

10.2.1.5 Quito paso

Finalmente, para optimizar nuestro programa nos centramos en la optimización del programa en LabView, mejorando el panel frontal, añadiendo un sistema de activación de clientes, consiguiendo de este modo poder observar únicamente los clientes que realmente se encuentran conectados. Inicialmente realizamos una lectura del número de clientes

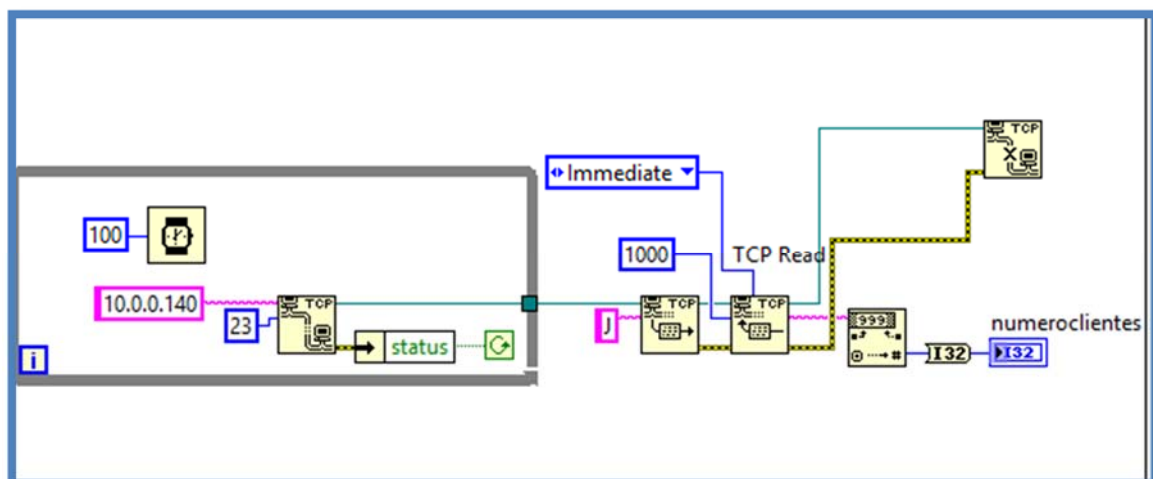


Fig. XXVI LabView Lectura Ethernet nº clientes

Como observamos en la figura anterior, utilizamos un bucle “while” y a la salida de la función de abrir canal de comunicación captamos su estado, y si es positivo, sabremos que se crea una conexión entre LabView y Arduino, asegurando que el sistema no nos dará un error, seguidamente realizamos la orden desde LabView para que Arduino nos envíe un dato específico, ya sea el valor de un cliente u otro, (realizando la escritura de J) y obteniendo una respuesta la cual leeremos con el bloque TCP Read, posteriormente realiza dos pasos, simultáneamente, por un lado cierra la conexión entre Arduino y LabView, mientras por otro, realizamos la conversión de la cadena de caracteres leídos a un valor numérico.

También introducimos un cambio de medida de temperatura, para poder observarla en grados Kelvin, grados Centígrados y grados Fahrenheit.

Inicialmente tendremos un frontal al iniciar nuestro programa como el que se muestra a continuación.

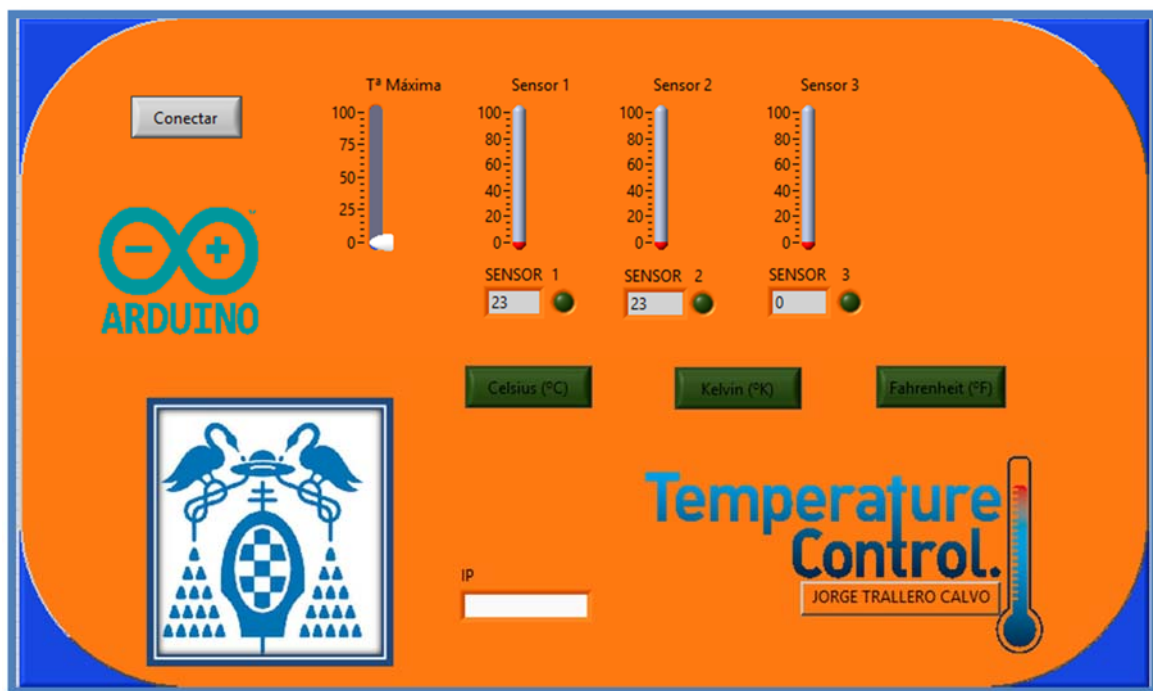


Fig. XXVII Panel frontal

10.2.1.6 Sexto paso

Por último realizamos la configuración para recoger todos los datos de todos y cada uno de los clientes, nuestra primera opción fue registrar los datos de manera continuada surgiendo un error como definimos en Lectura continuada LabView pág. 78, donde recogíamos el dato correcto únicamente en modo depuración, pero corriendo el programa únicamente mostrábamos valores erróneos, por lo que nuestro siguiente configuración se basó en la recogida de los datos cliente a cliente, creando una única conexión para cada cliente y de este modo obtener un consumo de energía menor.

La estructura utilizada fue del tipo “case”, donde enumerábamos varias conexiones diferenciando la escritura realizada por el LabView, que mediante un único contador realizábamos la vuelta necesaria dependiendo de los clientes registrados por el Arduino maestro, quien realizará una comunicación singular con el SCADA para informar el número de clientes que tiene registrados y de este modo activar o desactivar del panel frontal los indicadores que encontramos para cada cliente.

10.3 Desglose TOP-DOWN

10.3.1 Arduino cliente

El Arduino cliente es el encargado de registrar las temperaturas ambientales, en nuestro caso monitorizando la temperatura de una nave de almacenamiento de medicamentos, por ejemplo controlar la temperatura para una conservación adecuada de los medicamentos.

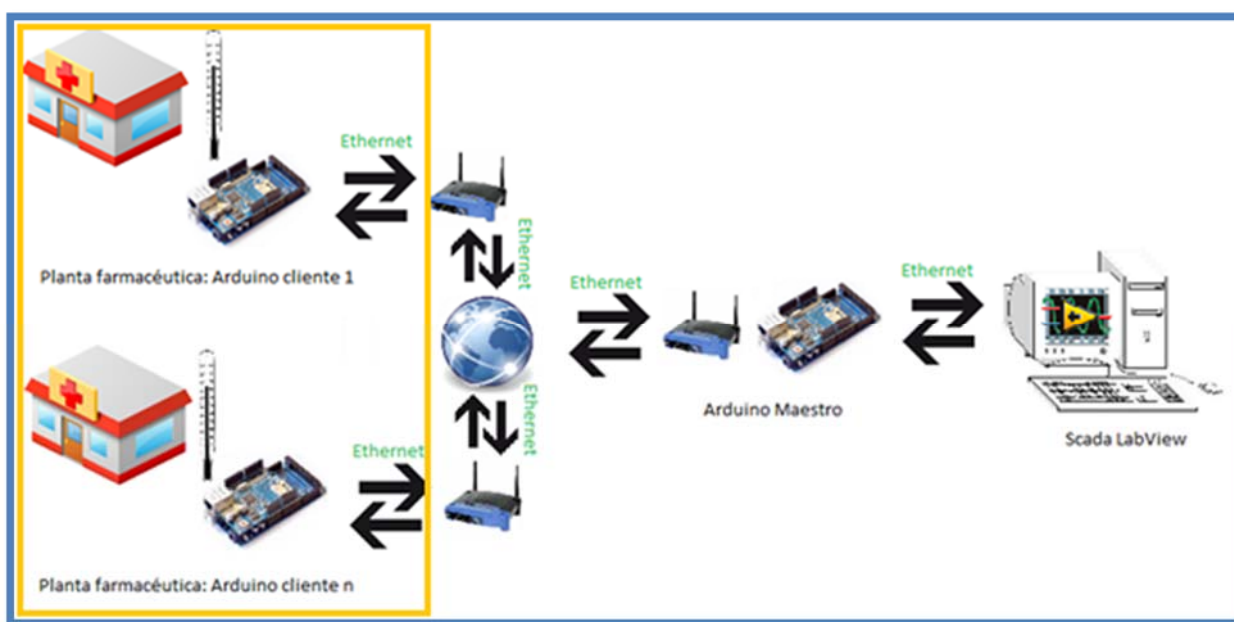


Fig. XXVIII Arduino cliente

En esta fase de montaje apreciamos que podemos contar con n Arduinos clientes, pudiendo ser n un número elevado de placas de adquisición de datos. Cada Arduino cliente se encuentra compuesto por un Arduino Mega 2560 (Fig. III) y por un Arduino Ethernet Shield (Fig. IV) siendo el segundo un escudo para poder realizar una conexión vía Internet. Su principal finalidad es registrar y enviar los datos obtenidos por el sensor DS18B20 (Fig. VI), esta adquisición de datos se realiza mediante los comandos que definidos en la base teórica (pág. 53). El sistema recoge la temperatura en grados centígrados, dicho valor lo guardamos en una variable global, para que mediante otra función creada, realice la conexión con el Arduino maestro y envíe de manera satisfactoria el valor obtenido.



Fig. XXIX Diagrama Arduino cliente

Como mostramos en el diagrama anterior, son los pasos que se repiten de manera cíclica que se producen en el Arduino cliente, comenzando en adquirir los datos pertinentes (en este caso la temperatura)

10.3.2 Arduino maestro

El Arduino maestro es el encargado de recopilar toda la información recibida de todos y cada uno de los Arduinos clientes, procediendo a su registro en variables globales y enviarlas mediante Ethernet a un controlador, en este caso el SCADA realizado mediante LabView.

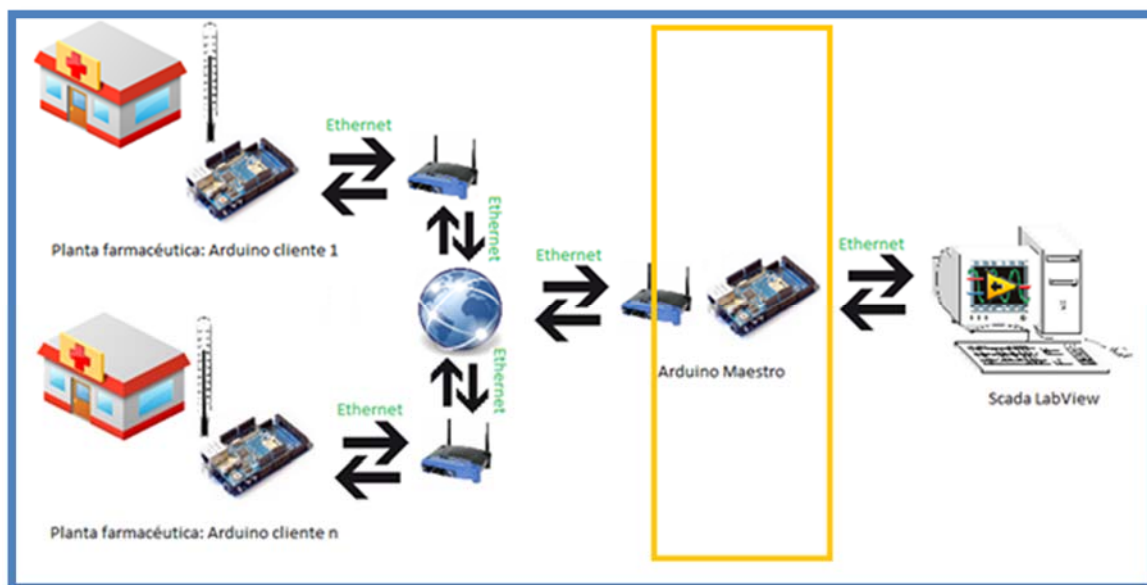


Fig. XXX Arduino maestro

Esta fase de montaje en la cual será la encargada de trabajar como “intermediario” entre los clientes y el SCADA para la visualización de todos los datos obtenidos se basa en un Arduino Mega 2560 (Fig. III) y Arduino Shield Ethernet (Fig. IV) el cual trabajara como escudo como con el sistema de cada cliente.



Fig. XXXI Diagrama Arduino Maestro

En este diagrama podemos observar los pasos seguidos por el Arduino maestro, en los tres primeros pasos (comenzando con la conexión con Arduino

cliente), esta enlazado con el diagrama mostrado en la Fig. XXIX, donde nos el Arduino se encarga de recoger los datos de cada cliente, uno a uno, cuando lee y registra todos los datos de cada uno de los clientes, realiza el cuarto paso que es la conexión con LabView, si hubiese algún problema el sistema envía un error, pero no se detiene, nuestro Arduino maestro seguirá intentando la conexión con el LabView durante un periodo de tiempo, que una vez superado comenzará de nuevo a recoger los datos de los clientes. Si el Arduino maestro realiza la conexión con LabView, realiza dos pasos, uno de lectura y otro de escritura, ya que el propio programa de LabView se encuentra implementado para enviar diferentes peticiones y recibir una respuesta correcta de cada cliente, nos referimos a que LabView deberá realizar una petición del dato requerido, es decir, dependerá de la información que queramos visualizar respecto a cada cliente, y seguidamente realiza la escritura del dato demandado por el sistema, por último cierra la conexión con LabView y continua de manera cíclica los pasos mostrados en el diagrama.

10.3.3 SCADA LabView

Por último, este bloque del proyecto es el encargado de la visualización y registro de todos y cada uno de los datos obtenidos.

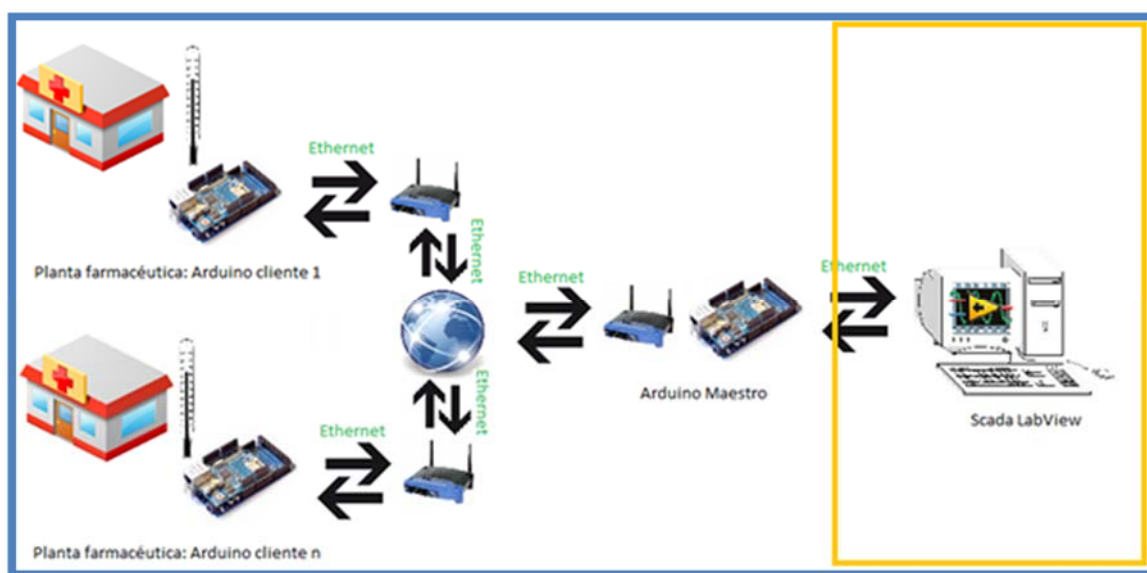


Fig. XXXII SCADA LabView

Para realizar esta funcionalidad este bloque realiza una conexión activa, como podemos observar en la Fig. XVI realizando peticiones de lectura al

Arduino maestro y que este realice una escritura del dato del cliente requerido, el cual será leído por el programa creado, registrado y mostrado por pantalla. Las operaciones seguidas son las mostradas en la Fig. XXXIII.



Fig. XXXIII Diagrama SCADA LabView

Este sistema realiza en primera instancia una escritura, realizando cuatro pasos específicos:

10.3.3.1 Conexión

La conexión realizada es activa como hemos comentado anteriormente (Fig. XVI), realizando una llamada a un sistema pasivo, siendo en nuestro caso el Arduino maestro, utilizando una dirección IP y un puerto en el cual se realiza la comunicación, en nuestro caso el 23 será el puerto definido para la comunicación entre Arduino maestro y LabView. El bloque principal que hace a una estación activa es el “TCP Open Connection”, cuyo icono es el siguiente.

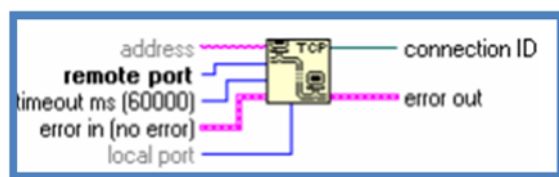


Fig. XXXIV TCP Open Connection

10.3.3.2 Escritura

El segundo paso a seguir, después de realizar la conexión con el nodo principal (Arduino maestro), será realizar la petición por lo cual se debe realizar una escritura, en nuestro caso un único carácter dependiendo del valor requerido. Este flujo de información entre la estación activa y la pasiva se controla con el bloque TCP write.

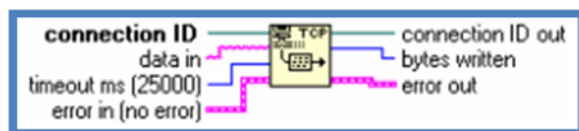


Fig. XXXV TCP Write

En “data in” introduciremos el carácter comentado anteriormente y el bloque consta de un timeout determinado por si no se realiza la escritura y de este modo se generará un mensaje de error en la salida de “error out”.

10.3.3.3 Lectura

En este paso LabView realiza una lectura del dato recibido desde el Arduino maestro obtenido a la salida una cadena de caracteres que posteriormente deberemos realizar la conversión de la cadena de caracteres (un dato tipo String) a un valor numérico DBL (double con un rango de valores numéricos comprendido en $2.23e-308 \leq |X| \leq 1.79e308$), y de este modo poder trabajar con ellos en el mismo entorno del programa. El bloque utilizado para realizar la lectura es el siguiente.

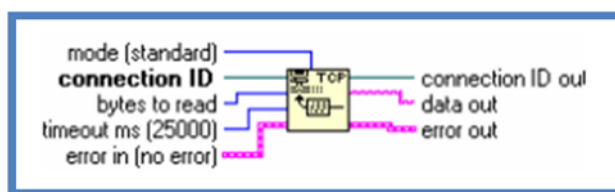


Fig. XXXVI TCP Read

Exactamente igual que en TCP Write, se producirá un error si el tiempo en la lectura tarda más de 25000ms. El modo de lectura será inmediato, que consiste que el sistema realizará la lectura hasta que no se reciba más bytes especificados en la entrada “bytes to read”.

10.3.3.4 Desconexión

Por último se debe cerrar la comunicación entre LabView y el Arduino maestro para que este pueda seguir escuchando por si necesita atender otras llamadas de algún otro cliente, y de este modo no generar ningún conflicto.

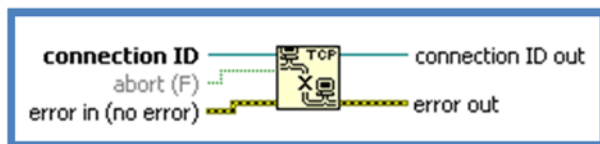


Fig. XXXVII TCP Close

El SCADA realizado es el mostrado en la Fig. XXXII SCADA LabView , detallando el interfaz completo, ya que si encontramos menos clientes el frontal no mostrará los clientes no existentes.

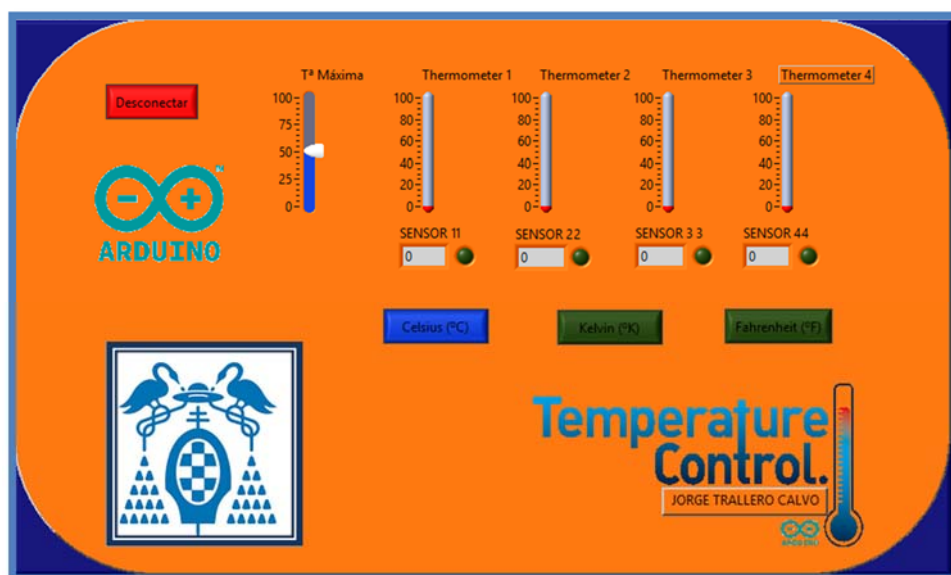



Fig. XXXVIII SCADA LabView

10.4 Arquitectura completa

En conjunto, el funcionamiento general sigue unas pautas regidas por el Arduino maestro, ya que es el encargado de escuchar a cada uno de los clientes, realizando las pertinentes conexiones y registrando a los nuevos clientes, desde este punto comenzará a realizar una conexión con el programa realizado en LabView pudiendo realizar una visualización cuando deseemos, mientras que la obtención de datos será de manera continuada. Inicialmente el sistema se encontrará sin ningún cliente, realizando únicamente un funcionamiento de

lectura del sensor de temperatura asignado para el Arduino maestro, de este modo si realizamos una conexión con el sistema de control en el ordenador obtendremos únicamente unos valores asignados a un sensor, el siguiente paso del sistema es la creación y conexión de nuevos clientes, que serán generados con un script específico en el cual únicamente se deberá modificar el mac para cada nuevo cliente, para de este modo identificarlos, y paso seguido será modificado por el Arduino maestro adjudicándole un nuevo puerto de conexión para recibir diversos valores.

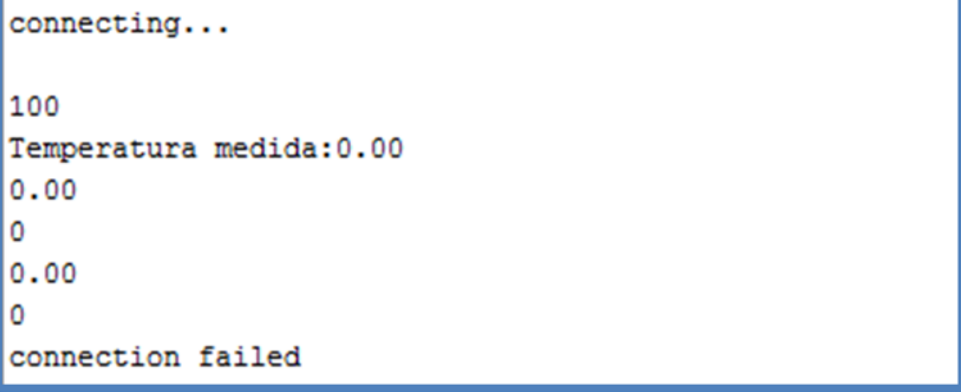
Inicialmente cuando nos encontramos sin ningún cliente podemos observar en pantalla desde Arduino lo siguiente:



```
Numero de clientes:  
0
```

Fig. XXXIX Puerto serie Arduino. 0 clientes

Como hemos realizado una conexión de clientes en caliente, es decir, que siempre nuestro sistema se encontrará funcionando y sin necesidad de pararlo, o refrescarlo, será capaz de encontrar clientes, los cuales estarán llamando al Arduino maestro. Mediante el puerto serie vemos el comportamiento de cada Arduino, si nuestro Arduino cliente está realizando la llamada al puerto 100 para realizar una conexión con el maestro, y éste no se encuentra disponible obtendremos la siguiente respuesta:



```
connecting...  
  
100  
Temperatura medida:0.00  
0.00  
0  
0.00  
0  
connection failed
```

Fig. XL Comunicación fallida

Si por otro lado el Arduino Maestro acepta esa llama proveniente del cliente obtendremos una comunicación entre los dos de la siguiente manera:

```
connected
Parece ser que tenemos otro cliente...
100
***Nuevo puerto:***
101
101

disconnecting...
```

Fig. XLI Reacción Arduino cliente

```
Numero de clientes:
0

NUEVO CLIENTE
DeNumero de clientes:
1
Client
1
connected.
Client
1
disconnected.
```

Fig. XLII Reacción Arduino maestro

Podemos observar en la Fig. XLI Reacción Arduino cliente que al realizar la llamada por el puerto 100, el Arduino maestro adjudica un nuevo puerto para controlar la lectura de temperatura de dicho cliente.

La comunicación con LabView, se realizará cliente a cliente, es decir, realizando conexiones diferentes para cada cliente, mostrando el Arduino maestro la siguiente información por pantalla.

Los datos obtenidos en LabView los mostramos de la siguiente forma:

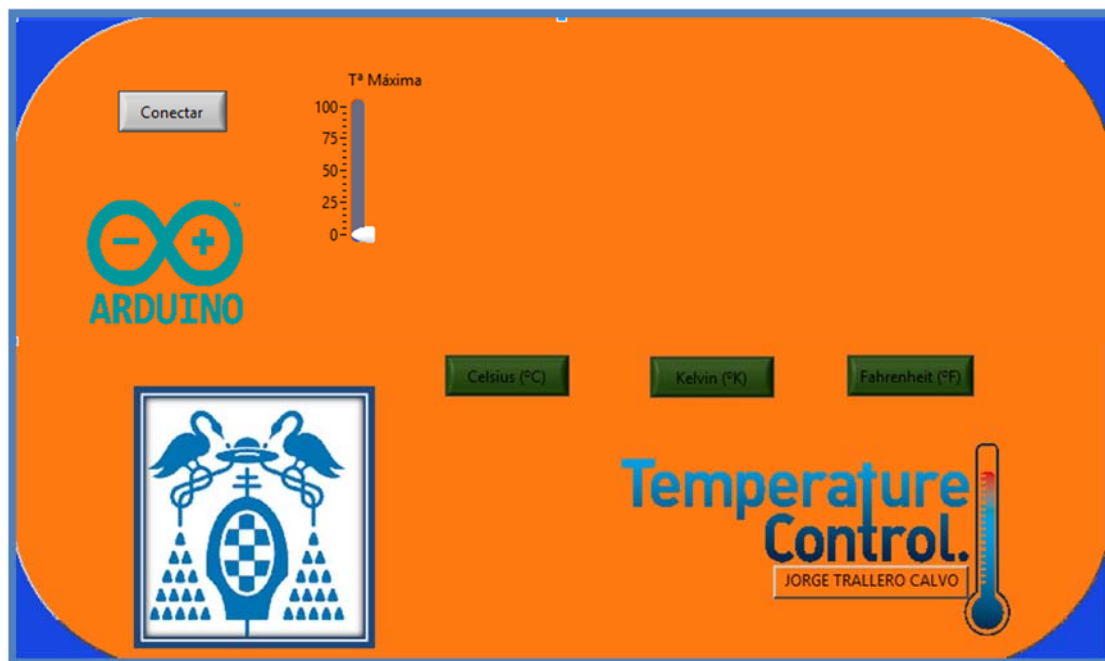


Fig. XLIII frontal LabView

Como observamos, inicialmente no encontramos ningún dato de ningún sensor, pero se introducirán respecto al número de clientes.

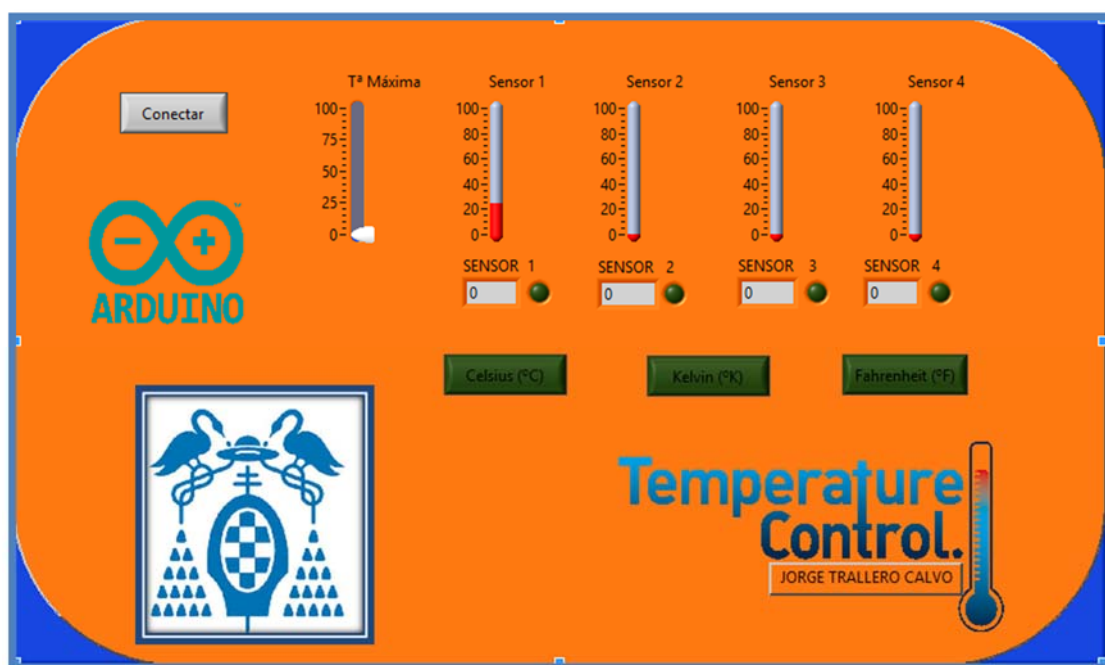


Fig. XLIV Frontal LabView 4 sensores activos

Finalmente mostrando el funcionamiento de tres sensores, observamos como el Arduino maestro trabaja de una consecutiva, enviando datos

empezando por los registrados por el mismo y continuando con los clientes, empezando desde el más antiguo al último cliente registrado.

Añadiendo el segundo cliente en caliente, observamos que lo admite y comienza a conectar con cada uno

```
NUEVO CLIENTE
DeNumero de clientes:
2
Client
1
connected.
Client
1
disconnected.
Client
2
connected.
Client
2
disconnected.
Client
2
connected.
Client
2
disconnected.
Client
```

Fig. XLV Comunicación cliente a cliente

Conectando a nuestro SCADA creado con LabView, los pasos seguidos por el Arduino Maestro son consecutivos como observamos en la siguiente captura.

```
Conectado a LabView
Temperatura medida:22.87
Client Labview conectado datos sensor maestro:...
22.87
Client Labview disconnected.

Conectado a LabView
Client Labview conectado datos sensor 1:...
23.31
Client Labview disconnected.

Client Labview conectado datos sensor 2:...
22.99
Client Labview disconnected.
```

Fig. XLVI Comunicación LabView

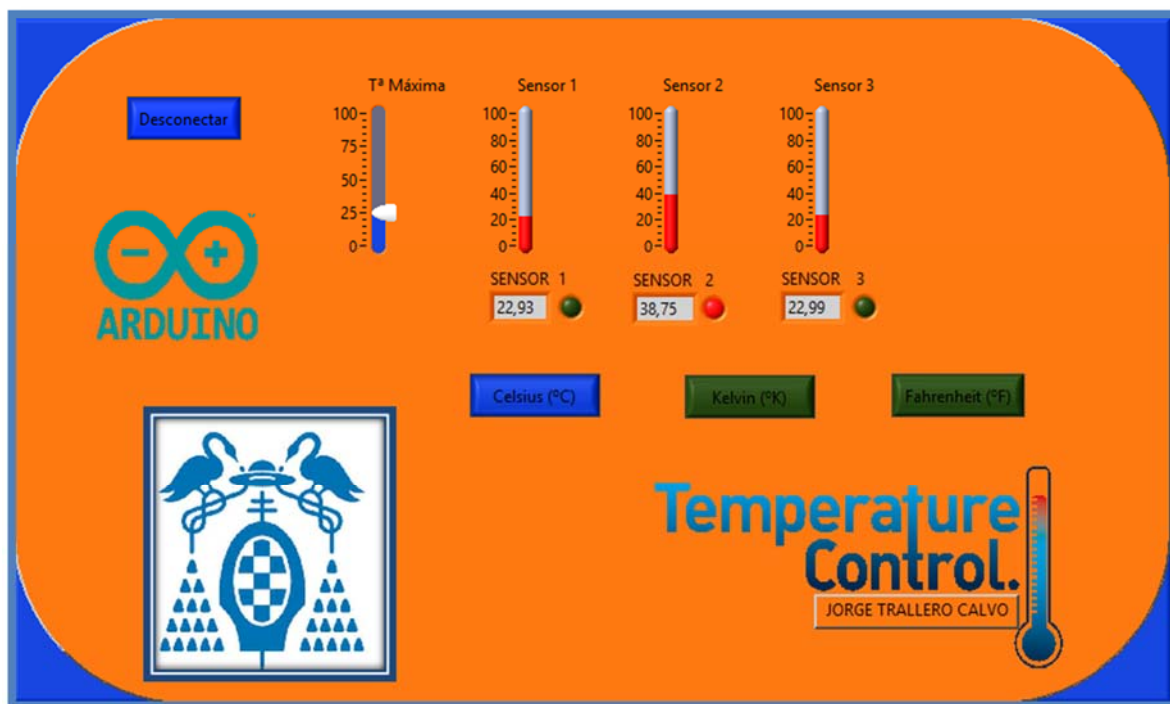


Fig. XLVII Resultado SCADA LabView

Como se puede observar conseguimos las tres temperaturas marcadas, y para el sensor 2 previamente calentado con una fuente de calor se activa la alarma en rojo ya que supera la temperatura máxima de 25 grados centígrados.

10.5 Problemas y soluciones

Realizando todo el montaje completo con cada apartado, surgieron varios problemas en cada paso, explicado los más importantes uno a uno y las soluciones tomadas.

10.5.1.1 Librería Ethernet

El dispositivo nunca abría un socket entre los dos sistemas conectados y la función "server.aviable" devuelve un valor negativo siempre hasta que el cliente no envía un dato, siendo un grave problema. La solución reside en suprimir la línea que se encuentra marcada en rojo del código correspondiente a la librería Ethernet, es decir, suprimimos la instrucción *if {}* pero no la sentencia que encontramos en su interior, ya que es necesario que devuelva el estado de client.

```
for (int sock = 0; sock < MAX SOCK_NUM; sock++)  
{  
    EthernetClient client(sock);  
    if (EthernetClass::_server_port[sock] == _port &&  
        (client.status() == SnSR::ESTABLISHED ||  
         client.status() == SnSR::CLOSE_WAIT))  
    {  
        If(client.available())// PROBLEMA DE CÓDIGO  
        {  
            return client;  
        }  
    }  
}
```

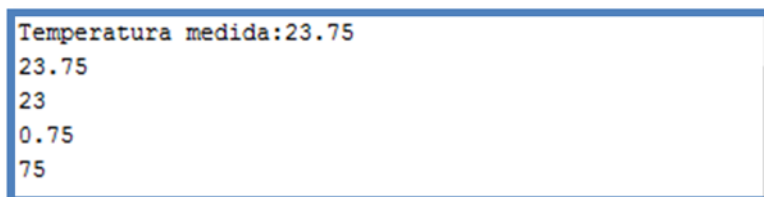
Como se observa en la parte del código que encontramos en la librería Ethernet, en la extensión .cpp. Las líneas de código recorren el bucle “for” cuatro veces, siendo el máximo permitido por el Arduino Ethernet Shield, pero al llegar a la línea remarcada en rojo, realiza una instrucción “if”, y únicamente la sentencia client.available() retorna un “1” si tenemos datos en el buffer, por lo que nos obliga a enviar un dato inicialmente para realizar el socket, por ello modificamos el archivo suprimiendo la sentencia “if”, pero manteniendo las instrucción de “return client;” y de esta manera poder crear el socket sin tener que tener datos en el buffer por obligación.

10.5.1.2 Comunicación Arduino cliente y maestro

Este problema a la hora de leer mediante el Arduino maestro, que únicamente leía la parte entera y la parte decimal siempre era rellenada con ceros, para solucionar este problema, a la hora de la comunicación, el Arduino cliente no enviaba únicamente un dato, sino enviamos dos, una parte entera y otra parte decimal, realizando la separación en el mismo sistema y en el maestro convierte la parte decimal dividiéndola entre 100 y sumada a la parte entera leída, solucionando este problema de manera satisfactoria pero sin llegar a comprender el fallo generado por la plataforma Arduino, siendo este uno de

muchos problemas a la hora de la programación y los fallos que aporta, muchas veces a causa de las librerías. Inconveniente comentado en la pag. 34.

En la Fig. XLVIII Separación parte decimal-parte entera mostramos mediante el puerto serie de Arduino como realizamos esa separación



```
Temperatura medida:23.75
23.75
23
0.75
75
```

Fig. XLVIII Separación parte decimal-parte entera

10.5.1.3 Lectura numérica LabView

No es realmente un problema, sino una limitación de LabView, ya que siempre que realiza una lectura se ha de realizar una conversión si queremos trabajar con la información de manera numérica ya que la lectura siempre será un string, realizando posteriormente una conversión DBL para poder realizar conversiones entre grados Kelvin, grados Centígrados y grados Celsius. A parte de la conversión, el dato leído no puede tener una parte decimal por lo tanto se ha de enviar el dato multiplicado por 100 y convertido con parte decimal de nuevo en LabView, de este modo nos será más sencillo poder realizar cálculos en LabView, para comparar temperaturas máxima o mínimas, y poder registrar los datos en una gráfica sin ningún problema añadido.

10.5.1.4 Lectura continuada LabView

Inicialmente la idea principal fue la lectura consecutiva de todos los valores de todos los clientes, surgiendo un grave problema a la hora de obtener los datos de manera correcta, ya que si no introducíamos unos tiempos muy elevados de espera entre lectura y lectura, muchos datos eran erróneos, leyendo “basura” y únicamente mostrando un valor correcto para el primer y último dato. Por ello se realiza una conexión diferente para cada dato leído, de este modo el tiempo entre lecturas es inferior y obtenemos una ventaja adicional, ya que cuanto menos tiempo de conexión, el consumo de energía será menor.

La primera configuración se realizó con un “Flat Sequence” como observamos en la siguiente figura

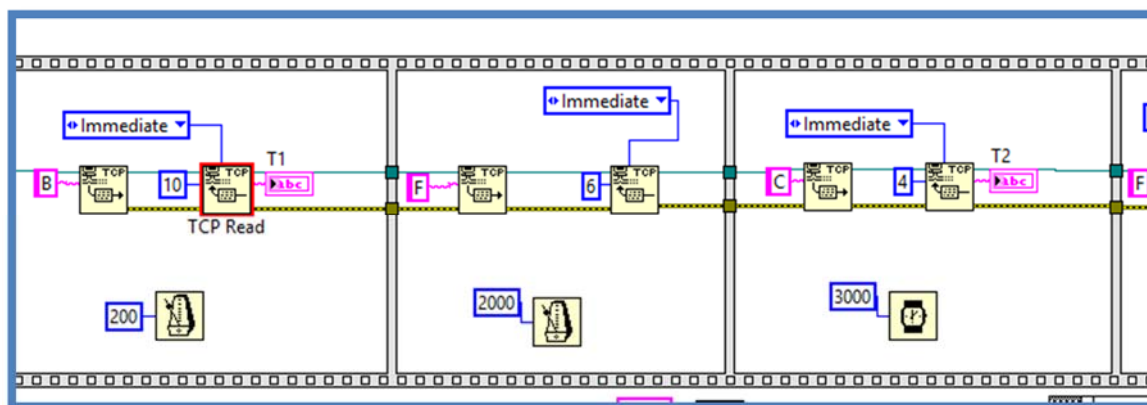


Fig. XLIX Flat Sequence Lectura

Y observa os que la mejor forma de realizar la lectura es la utilización de una estructura secuencial ejecutada de manera independiente para cada conexión entre el Arduino maestro y el SCADA.

11 Conclusiones y trabajos futuros

Como se recoge al inicio del proyecto, el objetivo del mismo consiste en determinar la forma de satisfacer a los clientes de un nivel salarial bajo con un sistema de adquisición de datos adaptado a sus fines de climatización, para un conjunto ,en nuestro caso, de almacenes de una pequeña farmacéutica mediante el uso de una tecnología de bajo coste. Para ello, se ha diseñado una instalación que reduce de forma efectiva el coste de tal desarrollo suponiendo que a corto plazo supone una amortización segura.

A continuación numeraremos las conclusiones más importantes del proyecto:

1-. La utilización del entorno Arduino proporciona una fácil configuración y de un coste mínimo, teniendo a nuestro alcance un gran volumen de documentación a causa de su crecimiento porcentual año tras año en su utilización por parte de pequeñas empresas y a nivel de usuario.

2-. Respecto al uso de LabView conseguimos crear una conexión segura entre dispositivos, sin proporcionar ningún error de recogida de datos ni visualización de los mismos, pudiendo asegurar un correcto funcionamiento respecto a este punto.

3-. Respecto a la inversión del proyecto, y observando el mercado, se podría asegurar un sistema rentable, y recuperar tal inversión en un corto periodo de tiempo.

4-. Al realizar un sistema de fácil utilización y bajo coste, creamos un sistema atractivo de cara al cliente, facilitándole un control de la climatización muy asequible.

5-. Como aspecto negativo, llegamos a la conclusión que en la utilización de placas Arduino no podemos asegurar un funcionamiento al 100% utilizando las librerías proporcionadas, ya que en algunas partes de estas librerías no se define bien algunos aspectos en su funcionamiento, pudiendo encontrar problemas a la hora de poner en marcha dicho sistema, teniendo que tener una gran atención a la utilización de dichas librerías sin revisarlas previamente.

Respecto al sistema creado, y a la hora de ir trabajando en la implementación de todas las partes, hemos llegado a barajear varios métodos para poder mejorar

la plataforma y darle continuidad pudiendo conseguir un proyecto más interesante y con un gran abanico de posibilidades, enfocando la continuidad del mismo enfocándolo desde varios puntos de vista, definiendo las siguientes propuestas.

11.1 Suprimir LabView

Inicialmente este proyecto gira en torno al Arduino y LabView, y la comunicación entre ellos, sería la implementación de un sistema sin la utilización del LabView, y de este modo quitarnos un problema ya sea de costo por la licencia, o de la disponibilidad de un sistema de control como un ordenador, de este modo se podría añadir una pantalla configurándola de tal forma que mostrase todos los valores de cada cliente, y poder así modificar el programa para un ilimitado número de clientes, y no como con LabView, que nos surge el problema de que únicamente se configura para un número de clientes limitados, y si requiriéramos más en un sistema, deberíamos implementarlos en el programa.

11.2 Implementar actuadores

Como hemos definido en nuestro proyecto, su finalidad es la monitorización de la temperatura de un recinto dedicado al mantenimiento de medicamentos, teniendo un control mínimo, a causa de que únicamente visualizamos la temperatura, pero también podríamos implementar unos actuadores, en este caso un refrigerador y un radiador por recinto, para que cuando la temperatura no se encontrase en un rango determinado, se activase uno u otro dispositivo.

11.3 Base de datos MongoDB y Hadoop

A mi punto de vista el trabajo futuro más interesante es la realización de dos bases de datos, una que realizase la adquisición y registro de datos durante un tiempo determinado, y otra como base historia del sistema, realizando un vuelco de la información semanal en esta base de datos, llegando a utilizar como una base de datos MongoDB siendo una base de datos NoSQL y de código abierto. Es una base de datos orientada a documentos, pensada para ser rápida, escalable y fácil de usar. Almacena estructuras de datos en documentos

tipo JSON con un esquema dinámico, donde podríamos determinar el tipo de medida, con la MAC del dispositivo, localización, etc.

Para la base histórica sería una manera efectiva utilizar Hadoop, es una infraestructura de programación de código abierto que permite la diseminación de datos en grandes clústeres de servidores utilitarios y que se procesen en paralelo. Utiliza un sistema de ficheros HDFS (Hadoop Distributed File System, Sistema Distribuido de Archivos Hadoop), es un sistema de ficheros distribuido que abstrae del almacenamiento físico y ofrece una visión única de todos los recursos de almacenamiento del cluster. Si un nodo del cluster se avería, el sistema continuará funcionando mientras es reparado utilizando la información replicada en otros nodos, es decir, HDFS se encarga de hacer múltiples copias de bloques de datos y de distribuirlos en múltiples nodos del sistema Hadoop. La parte complicada es que Hadoop es un sistema de almacenaje Big Data por lo tanto se necesita una gran memoria RAM y espacio suficiente para su correcto funcionamiento, y a la hora de realizar pruebas es una limitación para el proyecto.

12 Diagramas

En este apartado se describe el funcionamiento y finalidad de cada fichero creado para el fin de crear el sistema de adquisición de datos de temperatura.

12.1 Main_maestro

En el directorio E:\TFG_IoT_Jorge_Trallero\main_maestro\ encontramos el fichero **main_maestro.ino** donde su función bucle es la siguiente:

```
void loop()
{
    newclient();

    for( int x=lab; x<numero_clientes;x++)
    {
        lectura();
    }

    if(numero_clientes!=0)
    {
        for(int x=lab; x<20;x++)
        {
            LabView();
        }
    }
}
```

Fig. L void loop Maestro

Donde encontramos tres funciones importantes definiendo el fin de cada una:

12.1.1 Lectura():

Esta función realiza la lectura captada por el sensor digital de temperatura y registrando ese valor en una variable, para de este modo poder enviar posteriormente la información recogida al SCADA realizado con LabView

12.1.2 Newclient():

Esta función es la encargada de registrar si existe alguna llamada al puerto número 100, determinando si existe un nuevo cliente, justamente después de detectar esta llamada, realiza una conexión con el nuevo cliente asignándole un nuevo puerto para que las próximas comunicaciones con el Arduino maestro sean únicamente para enviar la información requerida.

12.1.3 LabView():

Función encargada de realizar las conexiones oportunas con el sistema producido mediante el programa LabView, e interactuar con él para enviarle la información, tanto del número de clientes existentes como las temperaturas registradas por cada uno.

12.2 Main_cliente

En el directorio **E:\TFG_IoT_Jorge_Trallero\main_cliente** encontramos el fichero **main_cliente.ino** donde su función bucle es la siguiente:

```
void loop()  
{  
    temperatura();  
    envio_datos();  
}
```

Fig. LI void loop Cliente

Donde encontramos dos funciones importantes, definiendo el fin de cada una:

12.2.1 Temperatura():

Esta función realiza la lectura captada por el sensor digital de temperatura y registrando ese valor en una variable, para de este modo poder enviar posteriormente la información recogida al Arduino maestro

12.2.2 Envio_datos():

Función implementada para la conexión con el Arduino maestro, diferenciamos dos partes, ya que si es la primera conexión y la llamada realizada es en dirección al puerto 100, el Arduino cliente pide un nuevo puerto para mantener comunicación con el Arduino maestro, una vez realizado este paso, con un nuevo puerto asignado, las siguientes comunicaciones serán únicamente para informar al Arduino maestro del nuevo valor de la magnitud registrada por el sensor digital de temperatura.

12.3 SCADATEMPERATURA.vi

Este archivo SCADATEMPERATURA.vi que encontramos en el CD de instalación en la ruta **E:\TFG_IoT_Jorge_Trallero\Project LabView** contiene el código fuente del programa creado para la visualización de los datos recogidos de las conexiones realizadas con el Arduino maestro.

Diferenciamos dos ventajas al abrir el archivo, el Front panel, siendo el aspecto final del ejecutable, y el block diagram, donde se encuentra todo el código.

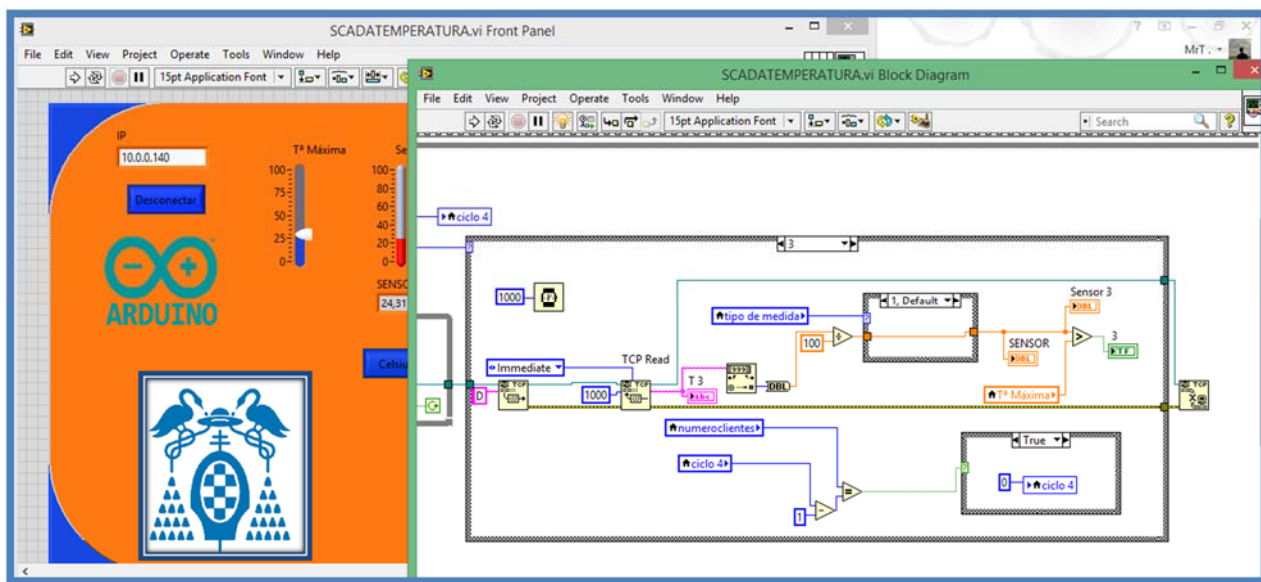


Fig. LII Diagram & Front LabView

13 Pliego de condiciones

13.1 Objetivo de este documento

El objetivo de este pliego es establecer las condiciones técnicas, económicas y administrativas del proyecto con el fin de que se realice en las condiciones específicas necesarias para su realización, definiendo dentro del pliego la especificación de los materiales, la reglamentación y normativas de obligado cumplimiento y por último, todos los aspectos del contrato que se refieren directamente al proyecto y que le afecten.

13.2 Condiciones generales

13.2.1 Documentos que rigen la ejecución de obra

A parte de las condiciones de este pliego, es de obligación las siguientes disposiciones de carácter general.

- 1-. Ordenanza de Trabajo para las Industrias de Producción [7]
- 2-. Ordenanza General de Seguridad e Higiene en el Trabajo [8]
- 3-. Reglamento Electrotécnico para Baja Tensión (2002) [9]
- 4-. Ley 21/1992, de 16 de julio, de Industria. [10]

13.2.2 Condiciones generales de pago y entrega del trabajo

1-. Las presentes Condiciones Generales de Pago y Entrega se aplicarán a todas las transacciones actuales y futuras mientras no se acuerde por escrito lo contrario.

2-. Se aceptaran entregas parciales siempre y cuando sea necesario.

3-. Todos los pagos se deberán realizar en un intervalo de un mes, después de la emisión de la factura, realizando un ingreso del 60% del total de la factura.

4-. La cuantía restante del proyecto será pagada después de la entrega.

5-. Según el contrato, el proyecto será propiedad del vendedor hasta que se deposite la totalidad de la deuda.

6-. Sólo se considerarán vinculantes los plazos garantizados por escrito.

7-. La penalización de por retraso en la entrega será de 1% mensual respecto a la facturación del proyecto, excluyendo de forma razonable los siguientes casos:

- Por cualquier modificación del cliente que altere el tiempo en la entrega
- Si por la parte del cliente el trabajo se demora por el trabajo a realizar por el comprador, demorando el cumplimiento de sus obligaciones.
- En caso de aparecer obstáculos que estén fuera de la voluntad del vendedor y se consideren casos de fuerza mayor.

8-. En cualquier caso si la entrega se demora más del 20% del tiempo estimado en la entrega, el cliente tendrá el derecho de reclamar la mercancía, en caso de poder comprobarse que el vendedor es culpable del retraso el cliente podrá renunciar al proyecto, teniendo derecho a una indemnización por los daños y perjuicios ocasionados.

13.3 Condiciones de materiales y equipos

13.3.1 Equipos de conexión

1-.Se han utilizados cuatro microcontroladores Atmega1280 de la marca Arduino, y cuatro escudos Ethernet implementado con los microcontroladores anteriormente nombrados, de la misma marca.

2-.En relación para la adquisición de la temperatura, disponemos de cuatro sensores ds18b20 de la marca DALLAS Semiconductor.

3-.Para realizar la correcta conexión entre los microcontroladores y los sensores se han utilizado cables jumper para una correcta conexión entre dispositivos.

4-.Por último se han utilizado cables Ethernet para la correcta conexión entre los microcontroladores e Internet, creando una red local con la utilización de un switch.

5-.Todos los materiales definidos son propiedad del Departamento de Electrónica de la Escuela Politécnica de la Universidad de Alcalá de Henares.

13.3.2 Equipos de programación

1.-Se ha utilizado un ordenador ordenadores para la elaboración de los programas tanto el encargado de la visualización de datos como los dos cargados en los microcontroladores.

2.-Para el programa desarrollado para la visualización se ha utilizado LabView 2012, mientras que para la programación de los microcontroladores se ha utilizado el propio de Arduino.

3.-El ordenador utilizado es propiedad del Graduado en Ingeniería Industrial en Electrónica y Automática Industrial, utilizando un procesador Intel® Core™ i5, pudiendo utilizar un procesador de gama inferior, y utilizando para su correcto funcionamiento un sistema operativo WINDOWS 7 como mínimo.

13.4 Condiciones de ejecución

1.-Para realizar una conexión correcta de todos los sensores utilizados, nunca deberemos tener una distancia superior a 20 metros entre el microcontrolador y el sensor.

2.-Para la exposición de este proyecto, se usará el material del Departamento de Electrónica de la citada Universidad.

3.-En la finalización del proyecto, todo el sistema será testeado y revisado por el Graduado Industrial en Electrónica y Automática Industrial.

14 Presupuesto

El presente presupuesto comprende los gastos de material, los gastos de mano de obra, el coste total de ejecución material, los gastos generales, el beneficio industrial, el presupuesto de ejecución por contrata y los honorarios.

14.1 Software

14.1.1 Listado de software necesario:

Software	Coste de licencia	Amortización	Tiempo utilizado	Total en €
Arduino 1.0.5	Gratuito	Indefinido	4 meses	0 €
LabView	995 €	1 año	4 meses	331,67 €
Office 2013	270 €	3 años	3 meses	22,5 €
TOTAL				354,17 €

14.2 Hardware

14.2.1 Listado de hardware necesario:

Componente	Precio unitario	Cantidad	Precio total
Arduino Mega 2560	18,31 €	4	73,24 €
Arduino Ethernet Shield	9,99 €	4	39,96 €
Cables Dupont	0,125€	40 (pack)	5 €
Sensor DS18B20	3,30 €	4	13,2
Entorno de montaje	1,5 €	4	6 €

PC Intel Core i5	1,5 € / Hora	415 h	622,5 €
TOTAL			759,9 €

14.3 Recursos

Realizando el cálculo final del coste que supone el conjunto de recursos utilizados:

Total de los recursos hardware	354,17 €
Total de los recursos software	759,9 €
TOTAL	1.114,07 €

El coste total de los materiales empleados asciende a **mil ciento catorce euros y siete céntimos de euro**.

14.4 Mano de obra

Control de tiempos	Horas
Investigación	20
Generación del código	150
Puesta en marcha inicial y Evaluación de errores	115
Manuales	60
Montaje e instalación	40
Puesta en marcha	25
Revisión y Finalización	5
TOTAL	415 h

Mediante la escala de sueldos refiriéndonos a un ingeniero como programador de autómatas, asciende a 21.000€ al año, al mes 1.750€ rondando a 10.94€ por hora.

Realización	Coste por horas	Horas	Total en euros
Diseño del proyecto	10.94	355	3.883,7 €
Realización de documentos	10.94	60	656,4 €
TOTAL			4.540,1 €

El coste total de la mano de obra asciende a **cuatro mil quinientos cuarenta euros y diez céntimos de euro.**

14.5 Presupuesto de ejecución material

Se calcula como la suma del coste total material y el coste total debido a la mano de obra.

Concepto	Coste
Material empleado	1.114,07 €
Mano de obra	4.540,1€
Presupuesto de ejecución material	5.654,17 €

El presupuesto de ejecución material asciende a **cinco mil seiscientos cincuenta y cuatro euros y diecisiete céntimos de euro.**

14.6 Presupuesto de ejecución por contrata

El presupuesto de ejecución por contrata es el importe a cobrar, añadiendo al presupuesto anterior, los gastos generales y el beneficio industrial. Los porcentajes para cada concepto no están fijados por ninguna normativa y utilizaremos los valores habituales, siendo un 13% para gastos generales y un 6% en concepto de beneficio industrial.

Concepto	Coste
Presupuesto de ejecución de material	5.654,17 €
13% Gastos generales	735,04 €
6% Beneficio industrial	339,25 €
Presupuesto de ejecución por contrata	6.728,46 €

El presupuesto de ejecución por contrata asciende a un total de **seis mil setecientos veintiocho euros y cuarenta y seis céntimos de euro.**

14.7 Presupuesto total.

Será el importe total a desembolsar, añadiendo al presupuesto de ejecución por contrata el impuesto sobre el valor añadido (I.V.A).

Concepto	Coste
Presupuesto de ejecución por contrata	6.728,46
IVA (21% P. ejecución por contrata)	1.412,98 €
TOTAL	8.141,44 €

El presupuesto total del proyecto asciende a **OCHO MIL CIENTO CUARENTA Y UN EUROS Y CUARENTA Y CUATRO CÉNTIMOS DE EURO.**

En Alcalá de Henares, a 20 de Junio de 2015.

Fdo.: Jorge Trallero Calvo

Graduado en Ingeniería Industrial en Automática y Electrónica Industrial

15 Montaje y manual de usuario

15.1 Introducción

Este documento detalla el procedimiento de instalación y montaje del sistema, diferenciando en configuración y arranque del proyecto.

Para empezar, nos centraremos en la parte hardware y los requisitos de cada uno de ellos.

Posteriormente, indicaremos el software base necesario para la plataforma para cada uno de sus componentes, así como una guía de instalación de referencia del mismo (No obstante, dependiendo del Sistema Operativo o de las preferencias del administrador de sistemas, este software base puede ser instalado de otra forma).

Para terminar, describiremos el proceso de instalación y configuración de cada uno de los clientes que componen el sistema y la forma en que se valida la instalación.

15.2 Recursos asociados a este documento

- **Arduino Mega 2560:** Microcontrolador encargado de registrar y manejarlos datos captados por el sensor DS18B20.
- **Arduino Ethernet Shield:** Escudo que permite la comunicación vía Ethernet entre los Arduinos Mega 2560 y el programa implementado en LabView.
- **DS18B20 Digital Temperature:**
Sensor de temperatura, encargado de captar la temperatura de su entorno de funcionamiento.
- **Programa Cliente Arduino:**

Programa realizado en lenguaje C++ determinado para la implementación relacionada con los dispositivos que trabajan como clientes.

- **Programa Maestro Arduino:**

Programa implementado en C++ determinado para el dispositivo encargado de conectar con los clientes y recibir y guardar sus datos y enviar dicha información a un SCADA que conectarán entre sí.

- **SCADA LabView:**

Aplicación de visualización de datos, implementada mediante LabView, capaz de mostrar mediante pantalla los datos recibidos del Arduino Maestro.

- **Librerías adicionales para arduino-1.0.5-r2:**

Es necesario incluir en las librerías del programa de código abierto de Arduino, la librería **dallas_temperature_control** adjunta en el CD de instalación en la ruta **E:\TFG_IoT_Jorge_Trallero\libraries**

15.3 Configuración de inicio

En este punto únicamente debemos configurar los archivos correspondientes a los Arduinos. El entorno de desarrollo es el siguiente:

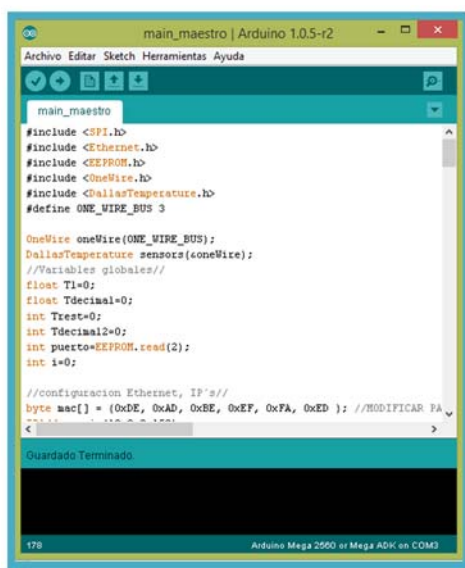


Fig. LIII Entorno de desarrollo Arduino

Donde para trabajar con el sistema deberemos realizar varias configuraciones:

1-.Escoger tipo de tarjeta en Herramientas\Tarjetas\Arduino Mega 2560

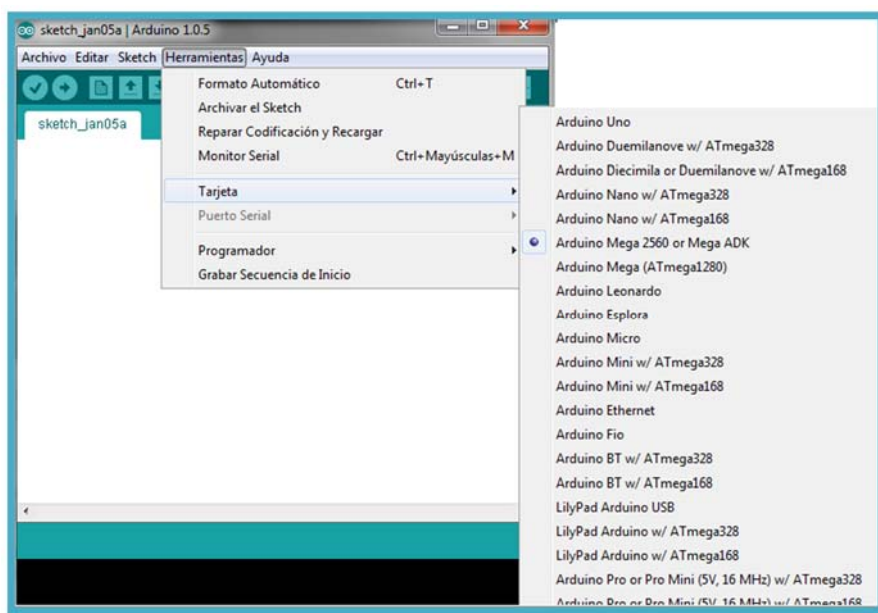


Fig. LIV Elección Tarjeta Arduino

2-.Definir puerto de conexión de la placa Arduino donde queremos realizar carga

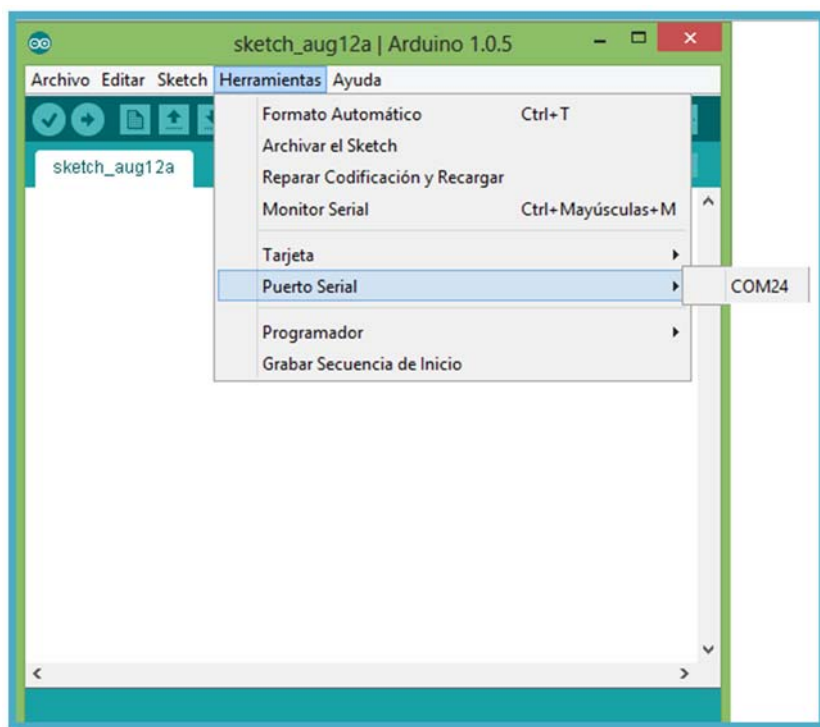


Fig. LV Elección Puerto de Conexión

3-.Realizar la carga correspondiente.

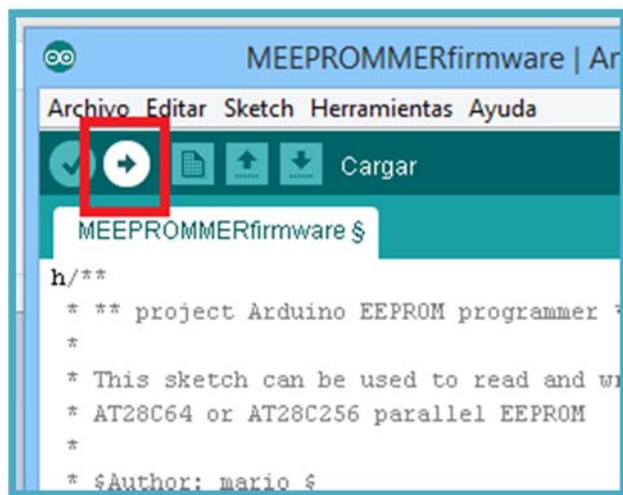


Fig. LVI Carga código Arduino

15.3.1 Configuración maestro

En la ruta **E:\TFG_IoT_Jorge_Trallero\main_cliente** encontramos el archivo **main_maestro.ino** correspondiente a la configuración del maestro donde se deben configurar varios apartados para su correcto funcionamiento.

Inicialmente deberemos configurar la IP de nuestro maestro, la dirección Gateway, la dirección de subnet y la dirección mac para su correcto funcionamiento. Todos estas modificaciones se deberán realizar en la cabecera del archivo **main_maestro.ino** como observamos en el siguiente fragmento de código.

```
/*configuración ethernet*/
byte mac[] = {0xDA, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

byte ip[] = { 10, 0, 0, 140 };
// the router's gateway address:
byte gateway[] = { 10, 0, 0, 1 };
// the subnet:
byte subnet[] = { 255, 255, 0, 0 };
```

Fig. LVII Configuración Maestro, paso 1

Una vez realizado este paso debemos realizar dos cargar en nuestro maestro.

La primera carga deberemos escribimos en una posición de la EEPROM un valor igual a cero, que es donde definimos el número de clientes inicial conectados a nuestro Arduino maestro.

```
void setup()
{
  EEPROM.write(3,0); //iniciar primera carga
  Serial.begin(9600);
  Serial.println("Numero de clientes:");
  Serial.println(numero_clientes);

  Ethernet.begin(mac, ip, gateway, subnet);
}
```

Fig. LVIII Configuración Maestro, paso 2

Una vez realizada esta carga, enmascaramos la sentencia de escritura en la EEPROM para que de este modo cuando conectemos el primer Arduino cliente empiece a contar número de clientes reales y no tengamos ningún cliente “parasito” de cargas anteriores.

Realizando una segunda carga enmascarando la escritura en la EEPROM para no encontrarnos con problemas si debemos resetear nuestro sistema.

```
void setup()
{
  //EEPROM.write(3,0); //iniciar primera carga
  Serial.begin(9600);
  Serial.println("Numero de clientes:");
  Serial.println(numero_clientes);
}
```

Fig. LIX Configuración Maestro, paso 3

Con este último paso ya disponemos de un maestro configurado para ser utilizado en nuestro sistema, una vez incluido en un entorno de funcionamiento no se deberá realizar ningún cambio más.

15.3.2 Configuración cliente

En la ruta **E:\TFG_IoT_Jorge_Trallero\main_cliente** encontramos el fichero **main_cliente.ino** correspondiente a la configuración de los clientes donde se deben configurar varios apartados para su correcto funcionamiento.

Inicialmente deberemos configurar la IP de nuestro cliente, la IP de conexión con el maestro y la dirección mac para su correcto funcionamiento. Todas estas modificaciones se deberán realizar en la cabecera del archivo **main_cliente.ino** como observamos en el siguiente fragmento de código.

```
//configuracion Ethernet, IP's//  
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFA, 0xED };  
IPAddress ip(10,0,0,150);  
IPAddress server(10,0,0,140); //IP del maestro
```

Fig. LX Configuración Cliente, paso 1

Como observamos en la figura la dirección IP del maestro es la que está definida en `IPAddress server(. . .)`;

Una vez realizado este paso debemos realizar dos cargar en nuestro Arduino cliente.

En la primera carga debemos guardar un valor de 100 en una posición de la EEPROM, en nuestro caso realizamos la escritura de '100' en la posición 2 (La posición 2 está definida en todo el fichero, si se desea modificar se debe modificar todas las escrituras y la lecturas de la EEPROM).

```
void setup()  
{  
  EEPROM.write(2,100); // para iniciar un nuevo cliente  
  Ethernet.begin(mac, ip);  
  Serial.begin(9600);  
  sensors.begin();  
}
```

Fig. LXI Configuración Cliente, paso 2

Una vez realizada esta carga, enmascaramos la sentencia de escritura en la EEPROM para que de este modo cuando conecte con el Arduino maestro le defina otro valor, siendo este valor el número de puerto al que se conectará nuestro Arduino cliente.

```
void setup()  
{  
  //EEPROM.write(2,100);// para iniciar un nuevo cliente  
  Ethernet.begin(mac, ip);  
  Serial.begin(9600);  
  sensors.begin();  
}
```

Fig. LXII Configuración Cliente, paso 3

Con este último paso ya disponemos de un cliente configurado para ser utilizado en nuestro sistema, una vez incluido en un entorno de funcionamiento no se deberá realizar ningún cambio más.

15.3.3 Conexión sensor de temperatura

Para realizar la conexión del sensor digital DS18B20, debemos realizar el siguiente montaje.

Los sensores utilizados disponen de tres cables, uno rojo (+Vcc), uno de color negro (GNG) y por último uno de color amarillo por donde envía la información. La conexión correcta se debe realizar con una resistencia de 47KΩ conectada entre la patilla roja (+Vcc) y la amarilla (datos) y posteriormente conectar nuestro sensor al Arduino.

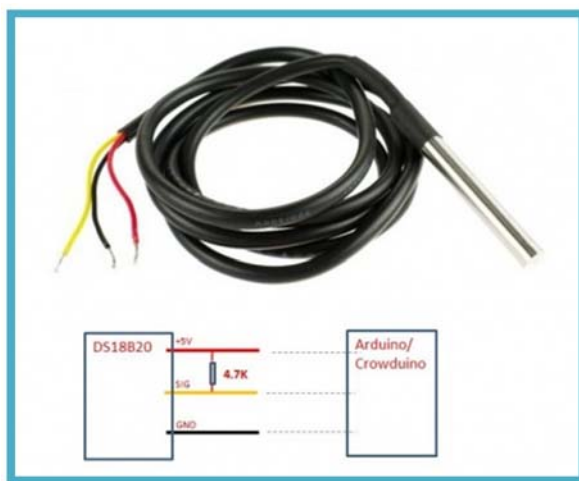


Fig. LXIII Conexión Sensor-Arduino

En nuestro montaje y como hemos configurado el sistema, debemos conectar el cable rojo a + 5V, en cualquiera de los puntos disponibles en nuestra placa, el cable rojo a GND, del mismo modo en cualquier punto GND de la placa, y por último, en nuestra configuración, el cable de datos de color amarillo deberá ir conectado al pin número 3 de nuestra placa.

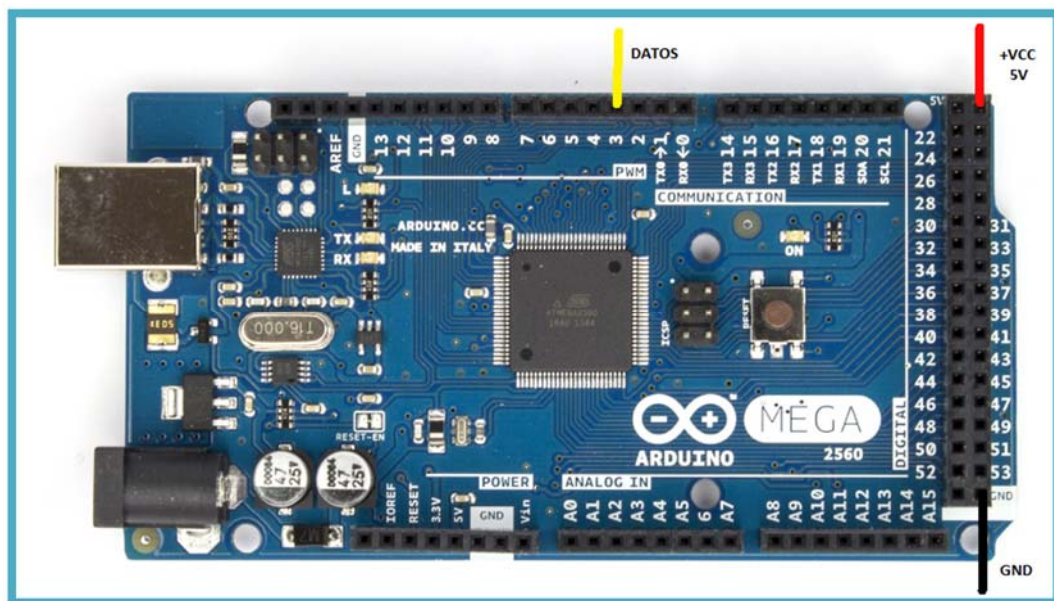


Fig. LXIV Conexión Arduino Mega

15.4 Arranque del sistema

Para poner en marcha nuestro sistema únicamente se deberá conectar inicialmente nuestro Arduino maestro, el cual se encargará de admitir a los nuevos Arduinos clientes que queramos conectar al sistema sin necesidad de realizar ninguna modificación.

Para visualizar el registro de temperaturas únicamente debemos iniciar el archivo ejecutable **TemperatureControl.exe** que encontraremos en la ruta **E:\TFG_IoT_Jorge_Trallero\builds\IoT_project**, el cual abriremos con un doble clic y se iniciará nuestro programa de visualización.

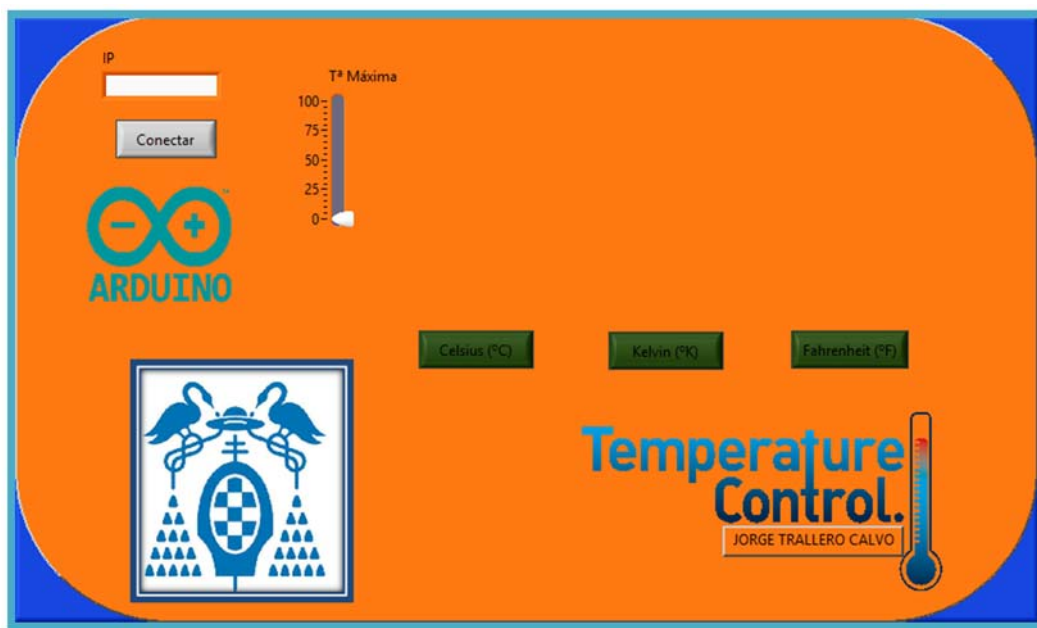


Fig. LXV Programa de visualización

Inicialmente no observamos ningún dato, ya que en el espacio donde se marca “IP” debemos introducir la IP configurada en nuestro Arduino maestro y posteriormente pulsaremos el interruptor “Conectar” para comenzar a recoger datos (1). También encontramos tres pulsadores más, los cuales nos servirán para modificar las unidades para mostrar la magnitud de temperatura (2). Por último tendremos la opción de modificar una temperatura máxima admisible, que si es superada por cada sensor, se activara una bombilla de alarma (3).

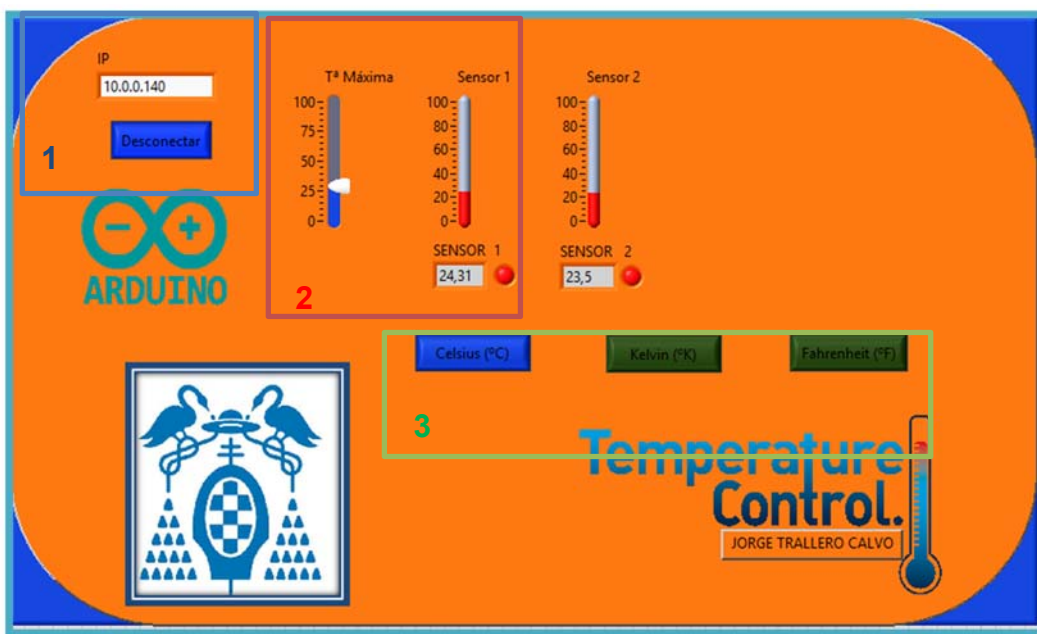


Fig. LXVI Programa de visualización, 1 cliente

16 Bibliografía

- [1] FI-WARE <http://en.blogthinkbig.com/2014/05/23/smart-cities-fi-ware/> (Junio2015)
- [2] Datasheet Arduino Mega 2560
<https://www.arduino.cc/en/Main/arduinoBoardMega2560> (Marzo2015)
- [3] Datasheet Arduino Ethernet Shield
<https://www.arduino.cc/en/pmwiki.php?n=Main/ArduinoEthernetShield>
(Marzo2015)
- [4] Arduino <https://www.arduino.cc/> (Febrero2015)
- [5] Carlos Fernando Jiménez “CONEXIÓN TCP/IP ENTRE DOS ESTACIONES USANDO LABVIEW 7 EXPRESS”
- [6] Librerías Arduino <http://arduino.cc/en/pmwiki.php?n=Reference/Libraries> (Enero2015)
- [7] Ordenanza de Trabajo para las Industrias de Producción
http://www.boe.es/diario_boe/txt.php?id=BOE-A-1970-954 (Junio2015)
- [8] Ordenanza higiene y seguridad laboral
http://www.boe.es/diario_boe/txt.php?id=BOE-A-1971-380 (Junio2015)
- [9] Reglamento Electrotécnico para Baja Tensión (2002)
<http://www.f2i2.net/legislacionseguridadindustrial/legislacionnacionalgrupo.aspx?idregl=76> (Junio2015)
- [10] Ley 21/1992, de 16 de julio, de Industria.
<http://boe.es/buscar/doc.php?id=BOE-A-1992-17363> (Junio2015)
- [11] IoT <http://www.startupsmart.com.au/technology/a-brief-history-and-future-of-the-internet-of-things/2014050912252.html> (Junio2015)

